EXPLORING THE EFFECT OF SAMPLING AND DIMENSIONALITY REDUCTION TECHNIQUES FOR INSIDER THREAT DETECTION

by

Keremalp Durdabak

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2024

This thesis is dedicated to my parents

Table of Contents

List of Tables

List of Figures

Abstract

Insider threats represent a significant challenge for organizations. They cost organizations money, time and resources. In 2024, a recent report by Code42 found that the average cost of an insider incident is $15 million. There are also costs to security teams, who are wasting time with limited resources. Thus, as artificial intelligence and machine learning has become mainstream, more and more security teams are looking to leverage these models to maximize their impact. This thesis explores a machine learning based approach in the field of insider threat detection with a specific focus on infiltration attacks. In particular, the impact of four dimensionality reduction and three sampling techniques are explored on the performance of machine learning models for detecting such attacks. These techniques are evaluated on three publicly available datasets using six ML models. The results indicate that in comparison to the original data features, it is possible to achieve comparable performances in detecting filtration attacks where dimensionality reduction is used. This capability potentially facilitates faster operational responses by reducing computational costs. The thesis research provides results and observations on the feasibility of utilizing reduced dimensionality for insider threat detection in filtration attack scenarios, presenting a foundation for further exploratory work in this field.

List of Abbreviations

DT Decision Tree

FS Fitness Sharing

GP Genetic Programming

GNB Gaussian Naive Bayes

ICA Independent Component Analysis

kNN K-Nearest Neighbour

LR Logistic Regression

NFS Non-Fitness Sharing

PCA Principal Component Analysis

RF Random Forest

SOCs Security Operation Centers

XGB XGBoost

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Dr. Nur Zincir-Heywood, for her support, guidance, and feedback throughout this research. Her expertise and encouragement have been essential to the completion of this work.

I am also grateful to Dr. Malcolm Heywood and Dr. Srinivas Sampalli for their valuable discussions and being my readers on my defense exam.

I would like to thank Dalhousie University, Mitacs, and OpenText for their support toward this thesis research and to everyone in the Network Information Management and Security (NIMS) research lab for their support and assistance. This work would not have been possible without them.

Chapter 1

Introduction

Insider threats represent a significant and complex challenge within the realm of cybersecurity. These threats are perpetrated by individuals within an organization—whether they be employees, contractors, or business partners—who have authorized access to the organization's systems and data. The motivations behind these actions vary widely, encompassing personal grievances, financial gain, or espionage, making them difficult to detect and mitigate. According to the CERT Insider Threat Center, insider threats are defined as threats originating from malicious or unintentional insiders whose authorized access to networks, systems, and data of an organization is exploited to negatively affect the confidentiality, integrity, availability, or physical state of the organization's information, information systems, or workforce [1].

The financial and operational consequences of insider threats are substantial. Organizations globally spend millions annually to address the fallout, which can range from severe data breaches to extensive operational disruptions. As reported by the Code42 in early 2024, the average annual cost of dealing with insider threats has escalated to $15 million [2], marking a significant increase from previous years [3].

Recent advancements in the field of cybersecurity have increasingly focused on the use of sophisticated machine learning techniques to combat insider threats. Research has explored a variety of approaches, including genetic programming, neural networks, and ensemble methods, each demonstrating significant potential in enhancing detection accuracy. Studies have utilized advanced feature reduction techniques, and various machine learning models across multiple datasets, revealing that tailored algorithms can effectively adapt to the dynamic nature of insider behaviors.

This thesis addresses the challenge of detecting two prevalent forms of insider threats: infiltration, which involves unauthorized access to systems to extract sensitive information, and exfiltration, which refers to the unauthorized transfer of information out of the organization. To combat these threats, the study explores the application of sampling, namely Stratified

sampling, Undersampling and Oversampling, as well as dimensionality reduction techniques, namely Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Genetic Programming (GP) with and without fitness sharing. These techniques are applied across three cybersecurity datasets: CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2 with different granularity levels of Week, Day, and Session. Additionally, six machine learning models, including Decision Tree (DT), Random Forest (RF), Gaussian Naive Bayes (GNN), K-Nearest Neighbour (kNN), Logistic Regression (LR), and XGBoost (XGB) are utilized to detect insider threats integrated with the sampling and dimensionality reduction techniques.

Thus, the primary objective of this research is to explore how these dimensionality reduction and sampling techniques can enhance (if at all) the performance of the aforementioned six machine learning models in detecting insider threats, thereby aiding Security Operations Centers (SOCs) and threat hunting teams. By reducing the dimensionality of data, these teams can potentially accelerate the detection processes, enabling quicker and more effective responses. This capability is crucial in reducing the time between threat identification and resolution, thereby minimizing the impact on organizational assets.

Moreover, the thesis provides a comparative analysis of different pipelines incorporating datasets, granularity levels, sampling techniques, and feature dimensionality reduction methods. This comprehensive approach allows SOCs and threat hunting teams to assess and choose the most effective technique — PCA, ICA, or GP—based — on their specific operational requirements. The insights gained from this research are intended to lay ground work for future approaches to enhance the effectiveness and efficiency of insider threat detection mechanisms.

Thus, in this research, my goal is to explore the effect of sampling and dimensionality reduction techniques on filtration attacks in the field of insider thread detection. To achieve this, I work on network traffic traces, transform them from traffic packets to traffic flows, ensuring data consistency and reliability — a key for the accurate training of machine learning models. I also examine the impact of various sampling methods, namely stratified sampling, oversampling, and undersampling. These sampling methods are essential for addressing class imbalances in the training data of machine learning models. These imbalances, if not corrected, can adversely affect the detection performance of the models. Through my findings, this thesis makes contributions to the field of cybersecurity by integrating multiple advanced techniques to enhance the insider threat detection. In particular, the novel contribution of

my research lies in exploration of the synergistic application of these techniques within a single investigative framework while specifically focusing on filtration attack scenarios for insider threat detection. My holistic approach aims to explore and leverage the strengths of the following to provide a more comprehensive and effective insider threat detection strategy:

- **Exploring Sampling Techniques:** Investigates the effects of stratified sampling, oversampling, and undersampling on machine learning models training and performance, focusing on insider threat detection and filtration attack scenarios.
- **Exploring the Dimensionality Reduction Techniques:** Evaluate the impact of PCA, ICA, and Genetic Programming (with and without fitness sharing) on the performance of machine learning models across various datasets—CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2.
- **Comparison of Reduced Feature Sets:** Compares the performance of machine learning models trained with reduced feature sets (e.g., PCA-5 through PCA-20 and ICA-5 through ICA-20) against models trained with original feature sets to assess the efficacy of dimensinality reduction.
- **Granularity-Based Analysis:** Provides insights into model performance across different data granularities —Week, Day, and Session—specifically using the CERT r4.2 dataset. This approach could enable SOCs and threat hunting teams to understand how data segmentation affects threat detection.
- **Scenario-Based Model Analysis:** Provides insights into performances of machine learning models under specific scenario combinations, specificially in the CERT r4.2 dataset (0vsAll and 0vs1_3), enhancing detection strategies for SOCs and threat hunting teams.

The remainder of this thesis is structured as follows: Chapter 2 reviews the literature pertinent to insider threat detection and the application of machine learning and dimensionality reduction techniques within this domain. Chapter 3 details the datasets utilized, the data preprocessing methods employed, and the specific machine learning models and dimensionality reduction techniques implemented. Chapter 4 presents the experimental setup and discusses the results obtained from the comparative analysis of the models. Finally, Chapter

5 concludes the thesis with a discussion of the findings and potential directions for future research.

Chapter 2

Related Work

This chapter examines the evolving landscape of insider threat detection within the field of cybersecurity. It critically reviews the development of detection methodologies, emphasizing the challenges and innovations that have shaped current practices. The focus is on how various strategies, from behavioral analytics to advanced computational techniques, have been employed to enhance the detection and mitigation of insider threats.

Specifically, this work underscores the important benefits these methodologies provide for Threat Hunting teams and Security Operation Centers (SOCs), facilitating more effective and efficient operations. Various machine learning approaches have become integral tools in this domain, supporting and refining cybersecurity methodologies by providing sophisticated data analysis capabilities that improve threat identification and response mechanisms.

A notable area of research within insider threat detection is the application of dimensionality reduction techniques, such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Genetic Programming-based Feature Selection (GP-FS). These techniques aim to reduce the complexity of the data while preserving the essential features that contribute to effective threat detection.

This chapter also reviews the implementation of sampling techniques, including stratified sampling, under-sampling, and over-sampling, to address class imbalance issues commonly encountered in cybersecurity datasets. Additionally, it highlights the importance of a granular approach to data analysis in providing detailed insights that can further enhance detection capabilities.

By synthesizing these methodologies, the chapter sets the context for the thesis research, which explores the combination of these advanced techniques within the specific scenario of insider threat detection in general, and infiltration attacks, in particular.

Le et al. [4] explored the adaptability of genetic programming to detect dynamic insider threats, showcasing how such methods can effectively handle the evolving nature of insider behaviors. They experimented on evolving GP on expanded feature space and evolving GP to recognize new malicious behaivours on the CERT r4.2 and CERT r5.2 datasets. The final results of the experiments were showing around 98% on the Normal Detection Rate (DR) whereas around 44% and 48% Insider Threat DR on 100 and 300 generations respectively. Overall, the results were comparable with the conventional machine learning algorithms such as LR, RF and Neural Networks.

Khan et al. [5] utilized neural networks to optimize hyper-parameter tuning for intrusion detection. Their proposed model achieved 99% accuracy on the CSE-CIC-IDS2018 dataset. The paper utilized DT based feature reduction and selection algorithm to obtain 19 features from the original 80 features.

Le et al. [6] analyzed how data granularity affects the performance of machine learning models such as LR, Neural Networks, RF and XGB in insider threat detection using the CERT r5.2 insider threat dataset. Among the models, RF performed the best with F1-score being higher than 75% on different granularity levels of the dataset. their research revealed that precision in data processing significantly impacts the effectiveness of models, advocating for a granular approach to data analysis to improve threat detection accuracy.

Raut et al. [7] provided a comprehensive review of the role of deep learning models such as Deep Belief Network, Autoencoder, Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) in detecting CERT Insider Threat Dataset. Among the models the paper utilized, RNN proved to be the most suitable solution with 95% AUC score due to its ability to process temporal log data.

Le et al. [8] introduced unsupervised learning ensembles such as Autoencoders, Lightweight Online Detector of Anomalies (LODA), Isolation Forest (IF) and Local Outlier Factor (LOF) to detect anomalies in insider threat scenarios on the CERT r4.2 and CERT r6.2 datasets. Autoencoders using percentile representation found to be the most useful with 20% investigation budget among day and week granularity levels with 90% and 98% AUC score for CERT r4.2 and r6.2 respectively.

Acharya et al. [9] explored the effectiveness of machine learning classifiers for network intrusion detection, utilizing the KDD99, UNSW-NB15 and CIC-IDS2017 datasets. Among the classifiers they used, Bagging and AdaBoost performed the best with close to 0% incorrect

classification rate and 1 AUC score.

Le et al. [10] explored the effectiveness of semi-supervised learning methods for insider threat detection under various label availability conditions. They employed three semi-supervised learning algorithms—Label Propagation, Label Spreading, and Self-Training—with the Self-Training method using a RF base classifier showing superior performance. Particularly notable were the results on the CERT insider threat test dataset (release 4.2), where the Self-Training method achieved an AUC of 0.992 and was able to detect 90% of malicious instances at a false positive rate of only 1%.

Karna et al. [11] proposed an ensemble-based filter feature selection technique to improve intrusion detection systems. By integrating multiple classifiers, their approach focused on enhancing the selection process for network security features, which significantly increased the detection rates of network anomalies. Their proposed model provided promising results in terms of accuracy by achieving 99.16% accuracy for the CIC-IDS2017 and 99.91% accuracy for the NSL-KDD datasets.

Pantelidis et al. [12] investigated the use of deep autoencoders and variational autoencoders for insider threat detection using the CERT r4.2 dataset. Their research emphasized the role of neural networks in extracting and analyzing complex data patterns. Their results showed that while Autoencoder performed slightly better in insider threat detection with 95% accuracy, Variational Autoencoder was overall proved to be the most effective in identifying threats with 96% accuracy.

Zheng et al. [13] introduced a novel approach using Dirichlet Marked Hawkes Processes for real-time insider threat detection. This method uniquely applied probabilistic learning to model the sequential behavior of users, enhancing the detection capabilities by focusing on temporal data patterns and anomaly detection with varying levels of base intensity λ_0 for the Hawkes process. Overall the best result of 88% AUC score has been achieved with $\lambda_0 = 0.1$ on the CERT insider threat dataset.

Li et al. [14] proposed a transformer based extreme semi-supervised framework (ESet) and achieved 97.60% F1-score on a dataset with only 1% of the instances being labeled. Their study demonstrated the use of semi-supervised models to effectively classify and predict different types of network intrusions, with varying ratios of labeled data on CIC-IDS2017 and CSE-CIC-IDS2018 datasets.

Xu and Zhou [15] explored byte pair encoding (BPE) algorithms for payload classification in network traffic, introducing a novel sub-word model that significantly enhances the precision of intrusion detection systems. Their methodology focused on feature extraction from payload data using the CIC-IDS 2017 dataset. The proposed feature extraction model applied on kNN, SVM and RF models shown comparable results with n-gram approach where both achieved around 99% F1-score.

Similarly, Erokhin et al. [16] addressed the challenge of feature selection for network intrusion detection, emphasizing the use of artificial neural networks and multilayer perceptrons to analyze the CSE-CIC-IDS2018 dataset. The paper applied Gini Index and Pearson Coefficient to select and reduce features, both achieving around 99% F1-score with only using around 20 features in contrast to 77 original features.

In another study, Guarino et al. [17] employed various machine learning techniques for early classification of network intrusions, utilizing a set of features pre-selected through a rigorous analytical process on the CSE-CIC-IDS2018 dataset. Among DT, RF, k-NN, GNB and SVM, RF gave the best results with $N_p = 10$ where N_p represents the initial packet amount of each biflow on the network.

Zhao et al. [18] developed a hybrid intrusion detection system that integrates feature selection and a weighted stacking classifier called (CFS-DE). By applying their proposed system on the NSL-KDD and CSE-CIC-IDS2018 dataset using RF, XGB and kNN, they demonstrated the proposed system's robustness in identifying and classifying network threats by achieving 88.25% F1-score on KDDTest+ and 99.88% F1-score on CSE-CIC-IDS2018 dataset.

Arshad et al. [19] introduced ML-IBotD, a machine learning-based intelligent botnet detection framework that utilizes traffic classification to identify botnet activities with high accuracy. Their work, which also used the CIC-IDS2017 dataset, showcased the application of SVM, kNN, DT and Ensemble Classifiers, whereas Ensemble Classifiers achieved an accuracy rate of 99.56%, emphasizing the potential usage of this framework against sophisticated cyber threats.

In his work, Turčaník [20] compares various evolutionary data clustering algorithms namely Genetic Algorithm, Particle Swarm Optimization and Differential Evolution for network attacks, using the CSE-CIC-IDS2018 dataset. Genetic Algorithm Clustering shown to obtain the lowest global distance from the correct instance in multi-dimensional search

space.

On the other hand, Singh and Chattopadhyay [21] explored hierarchical classification using an ensemble of feed-forward networks to analyze activity logs for insider threat detection in time-series data on different windows sizes. This approach enabled them to achieve around 99% AUC score on CERT r4.2 dataset, indicating the effectiveness of combining multiple neural network models to improve detection accuracy, especially in time-series data.

Bertrand et al. [22] proposed an unsupervised insider threat detection system based on Bayesian Gaussian Mixture Models, to detect insider threats based on user behavior using the CERT r4.2 dataset. The proposed approach competes with the state-of-the-art approaches such as LSTM-AutoEncoder and DBN-OCSVM with an accuracy and true negative rate of 93% Their research highlights the capability of unsupervised models to adaptively learn from data without predefined labels, offering a robust tool for real-time threat analysis.

Siregar et al. [23] presented a novel approach for optimizing the One-Class SVMs to identify cyber threats that are previously unknown. Their method utilized CIC-IDS2017 intrusion detection dataset and KBest for feature selection. The highest accuracy of 99% has been achieved with OCSVM with KBest-15 throughout different scenarios of the dataset.

It is important to note that, in addition to machine learning methods, significant research also employs non-ML approaches such as statistical and psychological methodologies for insider threat detection. For example, Padayachee [24] utilized opportunity theories from criminology to conceptualize insider threats and explore opportunity-reducing measures, while Greitzer et al. [25] developed a predictive modeling framework that incorporated psychological and motivational factors to identify potential insider threats. Furthermore, Guarino et al. [17] applied statistical methods, specifically Gaussian Naive Bayes, to enhance early classification in network intrusion detection. Additionally, Chadza et al. [26] employed Hidden Markov Models to predict sequential network attacks, further showcasing the application of statistical methods in this field.

In the following tables 2.1, 2.2, and 2.3 presents an overview of the aforementioned literature in terms of the machine learning methods, techniques utilized, and datasets employed. In the light of the literature I also utilize the CERT Insider Threat Dataset, CIC-IDS2017, and CSE-CIC-IDS2018. Additionally, I use feature extraction techniques such as GP, PCA, and ICA. Also, I employ a variety of machine learning models, including DT, RF, GNN, kNN, LR, and XGB. This exploratory approach aims to provide SOCs and threat hunting

teams with insights on optimizing model deployment for fast and effective insider threat detection.

Author	**Evolutionary Computation**	**Tree Based Algorithms**	**Ensemble Learning**	**Neural Networks**	**Probabilistic Learning**	**Instance-Based Learning**	**Regression Models**
Le et al., 2019	✓	✓	✓	✓	×	×	✓
Khan et al,. 2019	×	×	×	✓	×	×	×
Le et al., 2020	×	✓	✓	✓	×	×	✓
Raut et al,. 2020	×	×	×	✓	×	×	×
Le et al., 2021	×	✓	✓	✓	×	×	×
Acharya et al,. 2021	✓	✓	✓	×	✓	×	×
Le et al., 2021	×	✓	✓	✓	×	×	×
Karna et al,. 2021	×	✓	✓	×	✓	×	×
Pantelidis et al,. 2021	×	×	×	✓	×	×	×
Zheng et al,. 2021	×	✓	✓	×	×	×	×
Li et al,. 2022	×	✓	✓	✓	×	✓	×
Xu et al,. 2022	×	✓	✓	×	×	✓	×
Erokhin et al,. 2022	×	×	×	✓	×	×	×
Guarino et al,. 2022	×	✓	✓	×	✓	✓	×
Zhao et al,. 2022	×	✓	✓	×	×	×	×
Arshad et al,. 2023	×	✓	✓	×	×	✓	×
Siregar et al,. 2023	×	×	×	×	×	✓	×
Singh et al,. 2023	×	×	×	×	✓	×	×
Bertrand et al,. 2023	×	×	×	×	✓	×	×
Turčaník et al,. 2023	✓	×	×	×	×	×	×
Proposed Approach	✓	✓	✓	×	✓	✓	✓

Table 2.1: Summary of Machine Learning Methods used in the literature for insider threat detection

Author	**Dimensionality Reduction**	**Feature Extraction**	**Explainability**	**Model Comparison**	**Granular Approach**	**Sampling Techniques**
Le et al., 2019	×	✓	×	✓	✓	×
Khan et al,. 2019	✓	×	×	✓	×	✓
Le et al., 2020	×	✓	✓	✓	✓	✓
Raut et al,. 2020	✓	✓	×	✓	×	×
Le et al., 2021	×	✓	×	✓	✓	×
Acharya et al,. 2021	✓	×	✓	✓	✓	✓
Le et al., 2021	×	✓	×	✓	✓	×
Karna et al,. 2021	×	✓	×	✓	×	✓
Pantelidis et al,. 2021	×	×	×	✓	×	×
Zheng et al,. 2021	×	×	×	✓	✓	✓
Li et al,. 2022	×	✓	×	✓	×	×
Xu et al,. 2022	×	✓	×	✓	×	×
Erokhin et al,. 2022	✓	✓	×	×	×	✓
Guarino et al,. 2022	✓	✓	×	✓	×	×
Zhao et al,. 2022	✓	✓	×	✓	×	×
Arshad et al,. 2023	×	✓	✓	✓	×	×
Siregar et al,. 2023	✓	✓	×	×	✓	×
Singh et al,. 2023	✓	✓	×	×	✓	✓
Bertrand et al,. 2023	×	✓	✓	✓	×	×
Turčaník et al,. 2023	✓	×	✓	✓	×	×
Proposed Approach	✓	✓	✓	✓	✓	✓

Table 2.2: Characteristics observed in machine learning based approaches used in the literature for insider threat detection

Author	**CERT Insider Threat Dataset**	**CIC-IDS2017**	**CSE-CIC-IDS2018**
Le et al., 2019	✓	×	×
Khan et al,. 2019	×	×	✓
Le et al., 2020	✓	×	×
Raut et al,. 2020	✓	×	×
Le et al., 2021	✓	×	×
Acharya et al,. 2021	×	✓	×
Le et al., 2021	✓	×	×
Karna et al,. 2021	×	✓	×
Pantelidis et al,. 2021	✓	×	×
Zheng et al,. 2021	✓	×	×
Li et al,. 2022	×	✓	✓
Xu et al,. 2022	✓	×	×
Erokhin et al,. 2022	×	×	✓
Guarino et al,. 2022	×	×	✓
Zhao et al,. 2022	×	×	✓
Arshad et al,. 2023	×	✓	×
Siregar et al,. 2023	×	✓	×
Singh et al,. 2023	✓	×	×
Bertrand et al,. 2023	✓	×	×
Turčaník et al,. 2023	×	×	✓
Proposed Approach	✓	✓	✓

Table 2.3: Datasets that include infiltration attacks, are used in the literature, and publicly available

2.1 Summary

In this chapter, I review the application of machine learning based approaches to explore insider threat detection in cybersecurity. Building upon the previous works - as summarized in Tables 2.1, 2.2, and 2.3 - I have identified the research gaps and aim to address them in my research. Thus, this thesis explores the use of PCA, ICA, and GP-FS with and without fitness sharing, alongside six machine learning models across three different datasets, while implementing three sampling techniques, namely stratified sampling, undersampling, and oversampling. This comprehensive approach is tailored specifically for insider threat detection within two defined attack categories, namely infiltration and exfiltration attacks.

Chapter 3

Research Methodology

This chapter introduces the datasets, feature extraction methods, sampling strategies, and machine learning algorithms explored in this thesis.

3.1 Overview

Figure 3.1 provides an overview of the methodology used summarizing the process from dataset selection to the application of machine learning algorithms, encapsulating the methodology's core components and sequential steps.

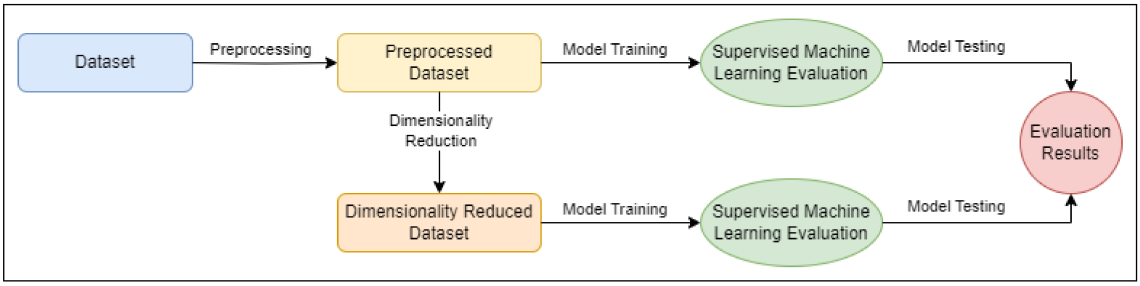

Figure 3.1: Overview of the methodology followed

To assess the effectiveness of dimensionality reduction in the context of insider threat detection, I engage with three key datasets: CIC-IDS2017, CSE-CIC-IDS2018, and a CERT r4.2 Insider Threat dataset at three different granularity levels of weekly, daily and session-based activities [6]. The analysis employed in this thesis incorporates a blend of feature dimensionality reduction techniques, including Genetic Programming (GP), Principal Component Analysis (PCA), and Independent Component Analysis (ICA). GP is examined both with and without the implementation of fitness sharing to understand its impact on feature reduction efficiency.

Further, I apply diverse sampling techniques, namely stratified sampling, oversampling, and undersampling to address potential imbalances in the datasets. These preparatory steps facilitate a more balanced training environment for the subsequent supervised learning phase.

In evaluating the transformed datasets, a suite of supervised learning models are deployed. These include DT (DT), RF (RF), GNB (GNB), K-Nearest Neighbors (kNN), LR (LR), and XGB (XGB). The goal is to investigate how the dimensionality reduction influences an ML model's performance. The comparison of results between models trained on the original versus the reduced-feature datasets aims to highlight the feasibility of achieving comparable performance with fewer features, thereby enhancing the computational efficiency and shortening the reaction time of Security Operations Centers (SOCs), and Threat Hunting teams within an organization.

3.2 Datasets

In this thesis, I investigate dimensionality reduction techniques to streamline the process of insider threat detection through machine learning. Central to this exploration is the analysis of datasets with varying scenarios of benign and infiltration activities. To this end, I meticulously selected three datasets for my study: CIC-IDS2017, CSE-CIC-IDS2018, and three granularity versions of the CERT r4.2 Insider Threat dataset. Each dataset encompasses distinct scenarios of benign activity and infiltration attempts. In doing so, the primary aim is to explore the performance of different features sets (using dimensionality reduction) across a variety of ML models compared to the original feature sets. This evaluation seeks to determine the feasibility of employing a reduced number of features without compromising the performance of the original datasets with more extensive feature sets. Given the considerable reduction in features from potentially hundreds to a mere handful, the implications for a threat hunting or machine learning team are important.

Understanding the composition and context of each dataset is crucial for this research. Therefore, the following sections will delve into the specifics of the CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2 datasets, outlining their scenarios and relevant metrics that will shape the analysis.

3.2.1 CIC-IDS2017

The CIC-IDS2017 dataset, designed by the Canadian Institute for Cybersecurity, is a comprehensive resource for intrusion detection research, containing a mix of benign and the most up-to-date malicious network activities that resemble real-world data. Captured over a week-long period in July 2017, this dataset features a variety of network behaviors including

normal operations and a series of sophisticated attacks such as DoS, DDoS, and infiltration, among others. This diversity allows for a rich analysis of network security measures under various attack scenarios. For this study, the Monday PCAP file, which represents benign activities, and the Thursday afternoon PCAP file which includes infiltration attacks, are specifically utilized to focus on the normal and attack vectors within network traffic, towards examination of dimensionality reduction in machine learning models for threat detection.

PCAP files	**Number of Packets**
Monday (Benign)	11,709,971
Thursday (Attack)	9,322,025

Table 3.1: CIC-IDS2017 PCAP files - Number of packets

3.2.2 CSE-CIC-IDS2018

The CSE-CIC-IDS2018 dataset, developed in a collaborative effort by the Canadian Institute for Cybersecurity and the Communications Security Establishment, extends the groundwork laid by its predecessor datasets by providing a more complex array of network behaviors, including benign operations and varied attack scenarios such as Brute-force, Heartbleed, and DoS attacks. This dataset captures the intricate dynamics of network traffic involving multiple departments and a broad range of networked devices, reflecting the challenges faced by contemporary intrusion detection systems. For this research, the dataset from Thursday-01-03-2018 is utilized, focusing on infiltration activities that simulate multi-tiered attack scenarios, thereby providing a detailed context for evaluating the effectiveness of dimensionality reduction techniques in identifying sophisticated cyber threats.

PCAP files	**Number of Packets**
Monday (Benign from CIC-IDS2017)	11,709,971
Thursday Part 1 (Attack)	198,804
Thursday Part 2 (Attack)	121,383
Thursday Part 3 (Attack)	462,067

Table 3.2: CSE-CIC-IDS2018 PCAP files - Number of packets

3.2.3 CERT r4.2

Evaluating insider threat detection systems is particularly challenging due to privacy concerns and the protection of organizational intellectual property. The CERT Insider Threat

Dataset, designed to mitigate these challenges, provides a simulated environment representing organizations with a detailed composition of employee activities. In this thesis, I focus on release 4.2 (CERT R4.2), which simulates a company with 1000 employees, encompassing 70 malicious insiders across three key scenarios: data exfiltration, intellectual property theft, and IT sabotage. Following Le et al.'s previous works [4, 6, 8], a granularity extractor script is built upon an open-source granularity extraction algorithm [1] to transform the CERT R4.2 dataset into subsets with weekly, daily, and session-based granularity, including statistically extracted features. This process allows for a more detailed analysis, with further discussion available in the preprocessing section.

Scenarios	**Description**
1-Data Exfiltration	A user, previously not using removable drives or working after hours, starts logging in late, uses a removable drive, and uploads sensitive data externally, leaving the organization shortly after.
2-Intellectual Property Theft	A user begins exploring job opportunities with competitors online and eventually steals data using a thumb drive at significantly higher rates than before, just before exiting the company.
3-IT Sabotage	A system administrator, after becoming disgruntled, deploys a keylogger to a superior's machine via a thumb drive, uses the logged information to cause organizational panic through a mass email, and exits the organization immediately.

Table 3.3: Summary of Insider Threat Scenarios in CERT r4.2 Dataset

Files	**Description**
Device.csv	Log of user's activity regarding connecting and disconnecting a thumb drive
Email.csv	Log of user's e-mail communication
File.csv	Log of user's activity regarding copying files to removable media devices
http.csv	Log of user's internet browsing history
Logon.csv	Log of user's workstation logon and logoff activity

Table 3.4: Description of CERT r4.2 Dataset Files

3.3 Data Engineering Workflow

In this section, I outline the preprocessing workflow that transforms raw datasets into refined data ready for analysis. For the CIC-IDS2017 and CSE-CIC-IDS2018 datasets, I begin by

[1] https://github.com/lcd-dal/feature-extraction-for-CERT-insider-threat-test-datasets

Attributes	Values
Duration	72 weeks (01/02/2010 - 05/16/2011)
# of Users	1,000
Scenarios (# of Mal. insiders)	1 (30), 2 (30), 3 (10)
# of Log on/off	854,860
# of Emails	2,629,979
# of Web	28,434,423
# of USB	405,380
# of File Access	445,581

Table 3.5: Overview of the CERT r4.2 dataset

procuring the PCAP files. These files are processed using Tranalyzer2[2], which converts them into flow data. I then filter out specific infiltration instances from the flow traffic, guided by timestamps. This subset is combined with benign instances to compile a comprehensive dataset. The datasets can be accessed from the Canadian Institute for Cybersecurity: CIC-IDS2017[3] and CSE-CIC-IDS2018[4] respectively. For the CERT r4.2 dataset, a granularity extraction script is employed to segment the data into daily, weekly, and session-based sets with statistically extracted features. The CERT r4.2 dataset can be accessed here[5].

3.3.1 Tranalyzer2

Tranalyzer2 is a sophisticated flow processing tool designed for decomposing network traffic captured in PCAP files into analyzable flows. It enhances network traffic analysis by providing detailed statistics and insights through its modular plugin architecture. In my thesis, I utilized Tranalyzer2 to convert the PCAP files from the CIC-IDS2017 and CSE-CIC-IDS2018 datasets into flow data, enabling the precise extraction of relevant features for analysis. Table 3.6 presents the Tranalyzer2 plugins used in this research. Using this default set of plugins, I initially obtained a flow aggregated dataset with 110 features, which were then meticulously refined to 77 by removing features that could potentially bias the target label, were redundant, exhibited no variability across the dataset, or contained null values for all its instances. Table 3.7 presents Tranalyzer2 Features used in this study.

[2] https://tranalyzer.com/
[3] https://www.unb.ca/cic/datasets/ids-2017.html
[4] https://www.unb.ca/cic/datasets/ids-2018.html
[5] https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247

Plugin	Description
basicFlow	Provides overall flow information
basicStats	Computes basic statistics of the flow
connStat	Statistics related to the connection state
icmpDecode	Decodes ICMP traffic details
macRecorder	Records MAC address information
portClassifier	Classifies flow based on port numbers
protoStats	Generates statistics based on protocols
tcpFlags	Analyzes TCP flags within the flow
tcpStates	Monitors the state of TCP connections
txtSink	Outputs flow data into a text file

Table 3.6: Tranalyzer2 Plugins used in this thesis

3.3.2 Preprocessing of CIC-IDS2017

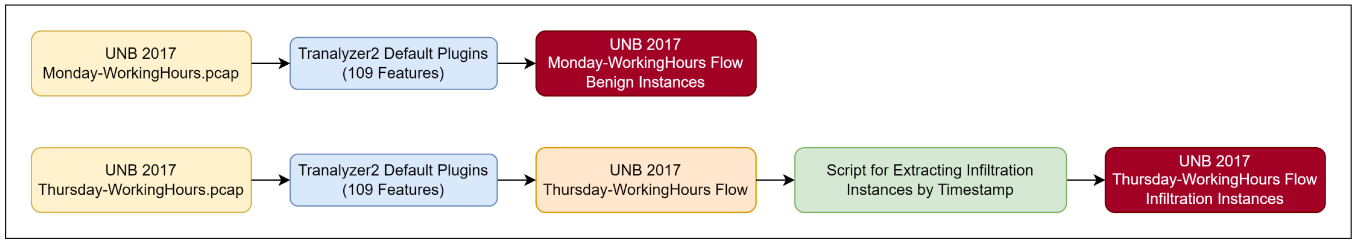

Figure 3.2: CIC-IDS2017 Dataset Flow Extraction

The preprocessing workflow for the CIC-IDS2017 dataset involves several steps as illustrated in Figure 3.2: initially, the CIC-IDS2017 Dataset was downloaded; subsequently, A flow aggregator of Tranalyzer2 was applied to the PCAP files for both Monday and Thursday sessions. Extracted Monday's flows were labeled as benign, while Thursday's flows underwent a process to extract the infiltration flows, guided by the timestamps provided on the dataset's website. Finally, both benign and infiltration flows were combined into a single dataset. Table 3.8 shows the number of flows for the CIC-IDS2017 dataset after employing Tranalyzer2 and preprocessing steps.

3.3.3 Preprocessing of CSE-CIC-IDS2018

For the CSE-CIC-IDS2018 dataset, as depicted in Figure 3.3: the Thursday PCAP files were processed with Tranalyzer2, from which infiltration instances were extracted and labeled based on the dataset's website timestamps. Owing to the absence of a benign dataset in

duration	numHdrs	hdrDesc	ethType
l4Proto	numPktsSnt	numPktsRcvd	numBytesSnt
numBytesRcvd	minPktSz	maxPktSz	avePktSize
stdPktSize	maxIAT	aveIAT	stdIAT
pktps	bytps	pktAsm	bytAsm
tcpFStat	ipMindIPID	ipMaxdIPID	ipMinTTL
ipMaxTTL	ipTTLChg	ipToS	ipFlags
ipOptCnt	ipOptCpCl_Num	ip6OptCntHH_D	ip6OptHH_D
tcpISeqN	tcpPSeqCnt	tcpSeqSntBytes	tcpSeqFaultCnt
tcpPAckCnt	tcpFlwLssAckRcvdBytes	tcpAckFaultCnt	tcpBFlgtMx
tcpInitWinSz	tcpAveWinSz	tcpMinWinSz	tcpMaxWinSz
tcpWinSzDwnCnt	tcpWinSzUpCnt	tcpWinSzChgDirCnt	tcpWinSzThRt
tcpFlags	tcpAnomaly	tcpOptPktCnt	tcpOptCnt
tcpOptions	tcpMSS	tcpWS	tcpTmS
tcpTmER	tcpEcI	tcpUtm	tcpBtm
tcpSSASAATrip	tcpRTTAckTripMin	tcpRTTAckTripMax	tcpRTTAckTripAve
tcpRTTAckTripJitAve	tcpRTTSseqAA	tcpRTTAckJitAve	tcpStatesAFlags
icmpStat	icmpTCcnt	icmpBFTypH_TypL_Code	icmpEchoSuccRatio
icmpPFindex	connF	connG	connNumPCnt
connNumBCnt			

Table 3.7: Tranalyzer2 aggregated 77 Flow Features used in this thesis

Flow Type	**Number of Flows**
Benign Flows	703,383
Infiltration Flows	181,571
Total Flows	884,954

Table 3.8: CIC-IDS2017 - Number of Flows

CSE-CIC-IDS2018, benign flows from CIC-IDS2017 were utilized. These elements were then combined into a single dataset for analysis. Table 3.9 shows the flow counts for the CSE-CIC-IDS2018 dataset after employing Tranalyzer2 and preprocessing steps.

Flow Type	**Number of Flows**
Benign Flows	703,382
Infiltration Flows	177,308
Total Flows	880,690

Table 3.9: CSE-CIC-IDS2018 - Number of Flows

3.3.4 Preprocessing of CERT r4.2

The preprocessing of the CERT r4.2 dataset involved several steps designed to enhance its utility for analyzing cybersecurity insider threats. A granularity extraction script was

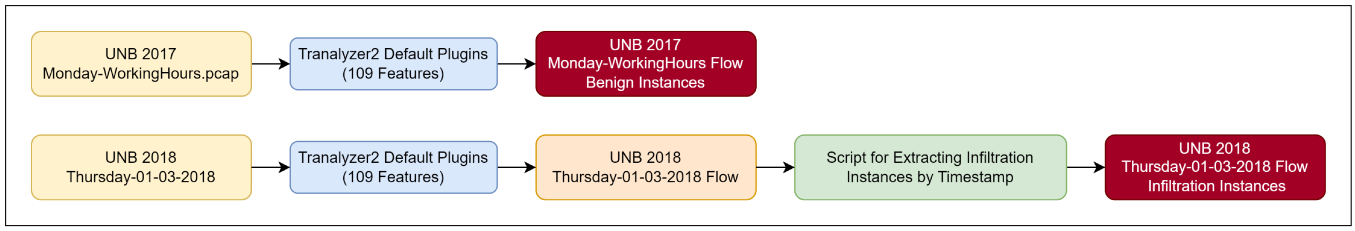

Figure 3.3: CSE-CIC-IDS2018 Dataset Flow Extraction

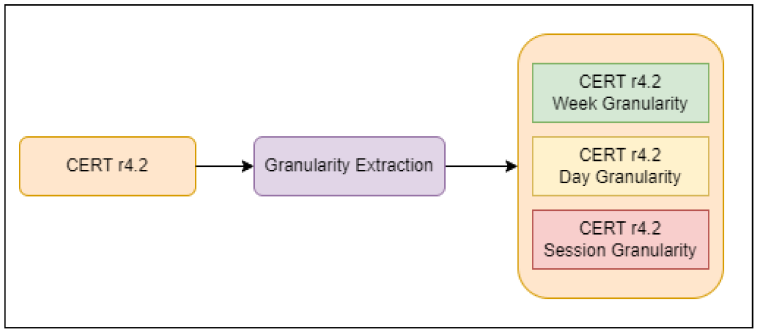

Figure 3.4: CERT r4.2 Flow Extraction based on different Granularities

employed to partition the dataset into three granularity levels: weekly, daily, and session-based. The process and results of this granularity extraction are effectively illustrated in Figure 3.4. Each granularity level provides a unique lens through which to examine user activities, as detailed in Table 3.10, where 'User-Week' refers to all user actions across all PCs over a week, 'User-Day' captures actions within a single day, and 'User-Session' encompasses all activities from login to logoff. Furthermore, Table 3.11 presents the distribution of data instances across various scenarios by these granularity levels, offering insights into the scope and scale of each scenario within the dataset.

Table 3.10: CERT r4.2 Extracted Granularity Levels

Data type	**Aggregation criterion**
User-Week	Week of user actions on all PCs
User-Day	Day of user actions on all PCs
User-Session	Session of user actions, from login to logoff on a PC

Table 3.11: Summary of the CERT r4.2 Scenarios

Feature Set	Normal	Sc 1	Sc 2	Sc 3	Number of Users
Week	66,840	52	254	10	1,000
Day	329,466	85	861	20	1,000
Session	469,497	69	1,013	32	1,000

3.4 Dimensionality Reduction

In this section, I provide the dimensionality reduction techniques applied to the datasets employed in this research. These include Principal Component Analysis (PCA) and Independent Component Analysis (ICA), implemented using the scikit-learn library. These algorithms serve to reduce the dimensionality of the proposed datasets, aiming to simplify the machine learning models without substantially compromising the performance. Additionally, I present a Genetic Programming (GP) based approach, which is a novel contribution of this research. In this case, GP is employed under two conditions: with fitness sharing (GP-FS) to promote diversity in the generated models (solutions), and without fitness sharing (GP-NFS), aimed at optimizing a specific objective function. This approach allows me to evaluate the impact of diversity in genetic programming solutions on the overall effectiveness and efficiency of the model in handling complex, multidimensional data. In doing so, the aim is to explore whether this approach would yield a dimensionality reduction and how it would affect the performance of the approach compared to the other dimensionality reduction techniques.

The number of features in the opriginal datasets used for training the machine learning models, without applying any further feature extraction or dimensionality reduction methods, is shown in Table 3.12. The number of features after applying various feature extraction or dimensionality reduction methods are shown in Table 3.13.

Table 3.12: Original Feature Counts for Each Dataset

Dataset	Feature Count
CIC-IDS2017	77
CSE-CIC-IDS2018	77
CERT r4.2 (Weekly)	667
CERT r4.2 (Daily)	507
CERT r4.2 (Session)	127

Table 3.13: Feature Counts after Applying Feature Extraction or Dimensionality Reduction Methods

Method	**Feature Count**
PCA/ICA-5	5
PCA/ICA-10	10
PCA/ICA-15	15
PCA/ICA-20	20
GP-FS	$\approx$ 8-15
GP-NFS	$\approx$ 10-15

3.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction while preserving as much variance as possible. It transforms a set of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This method is central to the study of multivariate data and continues to be a significant area of research within the field [27].

The process begins by standardizing the data matrix $\mathbf{X}$, where each variable has zero mean and unit variance. PCA seeks to identify the eigenvectors ($\mathbf{v}_i$) and eigenvalues of the covariance matrix Σ of $\mathbf{X}$, which are crucial in defining the new feature space. The covariance matrix is calculated as:

$$\Sigma = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X} \tag{3.1}$$

The principal components are then the directions along which the variance of the data is maximized. They are obtained by projecting the standardized data onto the eigenvectors. For instance, the projection of $\mathbf{X}$ onto the eigenvector corresponding to the largest eigenvalue gives the first principal component:

$$\mathbf{PC}_i = \mathbf{X} \cdot \mathbf{v}_i \tag{3.2}$$

In practice, the number of components retained is based on the amount of variance these components capture from the data. Typically, enough components are chosen to explain a high percentage of the total variance, simplifying data analysis without substantial information loss. PCA is extensively used in fields like genetics, finance, and image processing where reducing the number of variables is crucial without losing critical information.

3.4.2 Independent Component Analysis

Independent Component Analysis (ICA) is a computational method for separating a multivariate signal into additive subcomponents that are statistically independent or as independent as possible. This technique is commonly used in signal processing and data analysis to discover underlying factors or features [28].

ICA models the observed multivariate data as a linear combination of independent components. Mathematically, the model is described as:

$$\mathbf{X} = \mathbf{AS} \tag{3.3}$$

where $\mathbf{X}$ is the matrix of observed multivariate data, $\mathbf{A}$ is the mixing matrix, and $\mathbf{S}$ represents the matrix of independent components. The goal of ICA is to compute the unmixing matrix $\mathbf{W}$, which is the inverse of $\mathbf{A}$, to retrieve the independent components from the observed data.

The process of ICA involves several steps, including pre-processing to center and often whiten the observed data. This transforms the observed data so that their covariance matrix is the identity matrix, which simplifies the problem of finding the unmixing matrix. The core of the algorithm then iteratively adjusts $\mathbf{W}$ to maximize the statistical independence of the outputs computed as $\mathbf{WX}$.

The FastICA algorithm, as implemented in the scikit-learn library, is used in this research. It is an efficient and popular method for performing ICA. FastICA uses a fixed-point iteration scheme to find an estimate of the unmixing matrix $\mathbf{W}$. The algorithm maximizes the non-Gaussianity of the components, as measured by the negentropy, which provides a robust measure for statistical independence. It is particularly useful in scenarios such as signal processing (e.g., the cocktail party problem where multiple overlapping audio signals are separated), medical data analysis, and image processing. It excels in tasks where the hidden components are non-Gaussian and statistically independent from each other. The FastICA algorithm can be outlined as the following:

1. Center and whiten the observed data $\mathbf{X}$ to make it as Gaussian as possible, simplifying the unmixing process.

2. Choose an initial random weight vector $\mathbf{w}$.

3. Iterate over:

$$\mathbf{w}^+ = E\left\{\mathbf{X}g(\mathbf{w}^T\mathbf{X})\right\} - E\left\{g'(\mathbf{w}^T\mathbf{X})\right\}\mathbf{w} \tag{3.4}$$

   where g is a non-linear function, typically the logarithm of hyperbolic cosine or the exponential function.

4. Normalize the weight vector: $\mathbf{w} = \frac{\mathbf{w}^+}{\|\mathbf{w}^+\|}$.

5. Check for convergence and repeat until the components are maximally independent.

3.4.3 Genetic Programming

Genetic Programming (GP) is an evolutionary algorithm that evolves programs to solve specific problems. GP simulates the process of natural selection to breed a population of candidate programs through the use of genetic operators such as mutation, crossover, and selection. This method has been detailed in foundational texts that explore both its theoretical and practical applications [29].

In GP, a population of programs is evolved over a series of generations. Each program is evaluated for its performance on a given task, with better-performing programs being more likely to be selected as parents for the next generation. Through the application of crossover and mutation, new programs —'children'— are created from the 'parents'. These children then replace some of the less fit programs in the population, a process known as 'replacement'.

The cycle of selection, crossover, mutation, and replacement continues until the termination criteria are met, which often involves finding a program that meets the performance requirements or reaching a maximum number of generations. GP is particularly adept at discovering solutions to problems that are hard to define explicitly in traditional programming terms.

In this thesis a Linear Genetic Programming (LGP) based apporach is employed. The LGP is an adaptation of the GP paradigm that evolves linear sequences of instructions, similar to an assembly language, rather than tree-based structures [30]. This approach uses a set of registers to store intermediate values during the execution of the program. In LGP, each individual in the population is a computer program composed of a list of imperative instructions, where each instruction operates on one or more registers and possibly on constants from a predefined set.

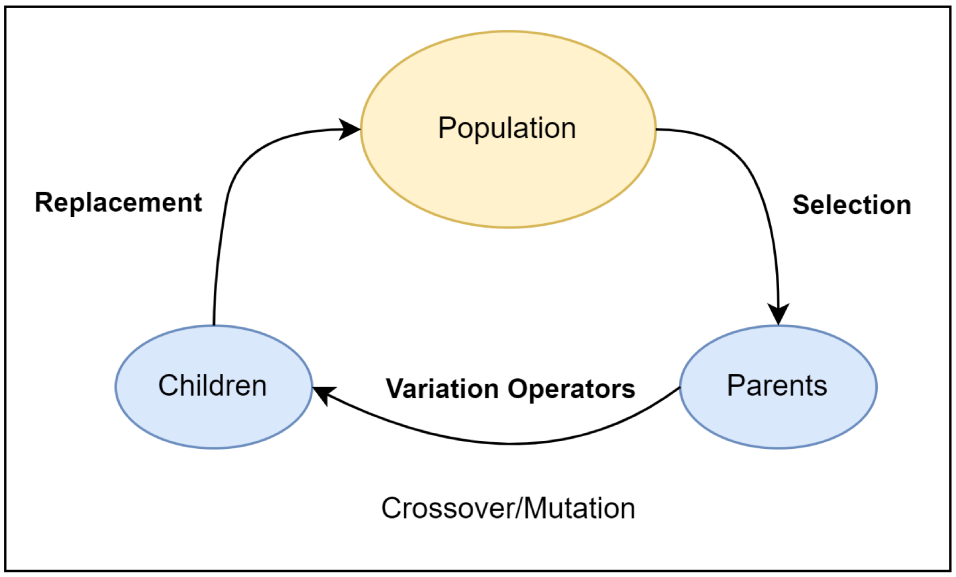

Figure 3.5: GP (LGP) Evolution Cycle Overview

The evolution cycle of LGP is depicted in Figure 3.5, which shows the flow of genetic operations. The population, consisting of multiple linear programs, undergoes a selection process where individual programs are chosen based on their fitness levels. These programs are referred to as 'parents'. Through genetic operators, such as crossover and mutation, new candidate solutions, called 'children', are created. Crossover involves the exchange of instruction sequences between parent programs, while mutation alters one or more instructions in a program. Once new children are produced, they may replace existing programs in the population, depending on their fitness and the replacement strategy employed. This cycle of selection, variation, and replacement continues for many generations until a termination condition is met, which may be a satisfactory level of fitness or a maximum number of generations.

The application of GP in this thesis is further detailed in the procedural flowchart shown in Figure 3.6. This diagram elucidates the step-by-step process of the GP algorithm tailored to dimensionality reduction. The process starts by generating multiple populations, each tailored to predict a unique target label from the dataset. Within these populations, teams of individuals evolve over 1000 generations, enhancing their capability to predict the specific target label instances. Fitness Sharing is employed to foster diversity within the population and to mitigate premature convergence on sub-optimal solutions. The fitness of each team

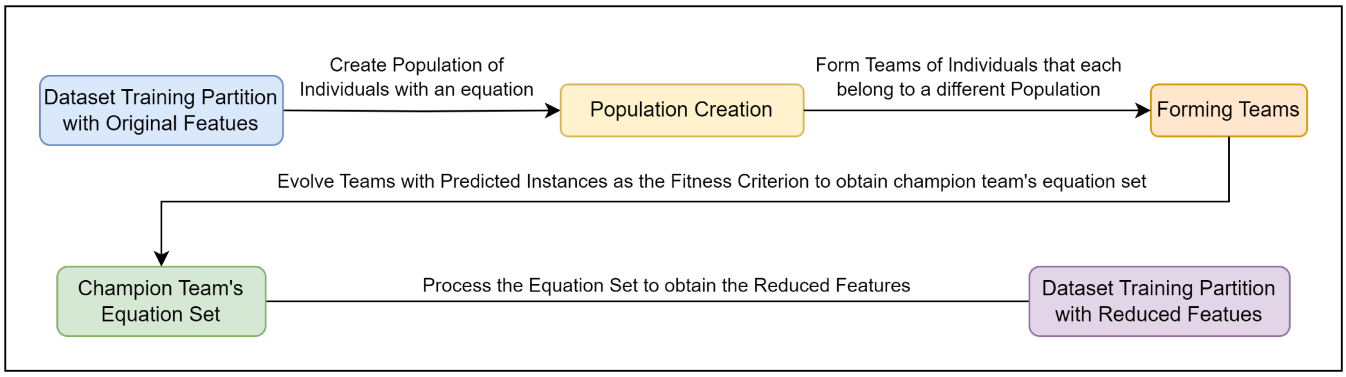

Figure 3.6: GP (LGP) Algorithm Overview

is gauged by their accuracy in predicting instances, and the team with the highest instance classification score across 20 iterations—utilizing both Fitness Sharing and No Fitness Sharing strategies—is deemed the champion. The equations from the highest scoring champion team are then used to derive a final reduced feature set, transforming the original training dataset into a GP-reduced training feature set.

Individual and Population Generation

The generation of individuals and populations within the (L)GP system adheres to a parameter-driven approach. Multiple populations are generated in parallel, each mapping to a unique target label within the dataset. The total number of populations is equal to the number of unique target labels. Moreover, each individual is characterized by an equation consisting of a series of operations—$+, -, \times 2, \div 2$—applied to operands from the dataset's feature set. The multiplication and division are specifically by two to ensure computational simplicity and prevent undefined operations, such as division by zero. For these operations, a single operand from the feature set is selected, denoted generically as F_i, where i is the feature index. In contrast, addition and subtraction require two operands, hence two distinct features, F_i and F_j, are chosen for each operation. An example equation belongs to an Individual can be represented as:

$$I_p = (F_{49} \times 2) + (F_{49} - F_2) + (F_{72} \div 2) + (F_{49} \times 2) + (F_1 + F_{38}) \tag{3.5}$$

An individual's equation is formed by concatenating randomly selected operations with their corresponding operands, repeated until the predetermined operation count is met. This

count is specified by a parameter set prior to execution. For a dataset with N features, the feature pool is $\{F_0, F_1, \ldots, F_{N-1}\}$, and the selection process is random with respect to the operations and the associated operand count.

Team Formation

Following the initialization of populations, the (L)GP model orchestrates a team formation strategy as seen in Figure 3.7. Each team is an ensemble of individuals, where every individual represents a unique solution vector from a distinct population. These teams are engineered to encapsulate the multifaceted nature of the classification task, with one individual from each population per team. This team formation strategy is based on the methodologies developed by Brameier et al. [31], and further explored by Doucette et al. [32] in genetic programming.

In the context of my model, a team T_0 consists of individuals $P_0, P_1, \ldots, P_n$, where n corresponds to the number of unique target labels in the dataset. Each team is assessed on its collective ability to predict instances across all target labels, with the fitness score of the team being derived from the highest performing individual within the team for each dataset instance. This methodology leverages the strengths of diverse individuals, fostering a competitive yet collaborative environment where each member's contribution is vital for the success of the whole.

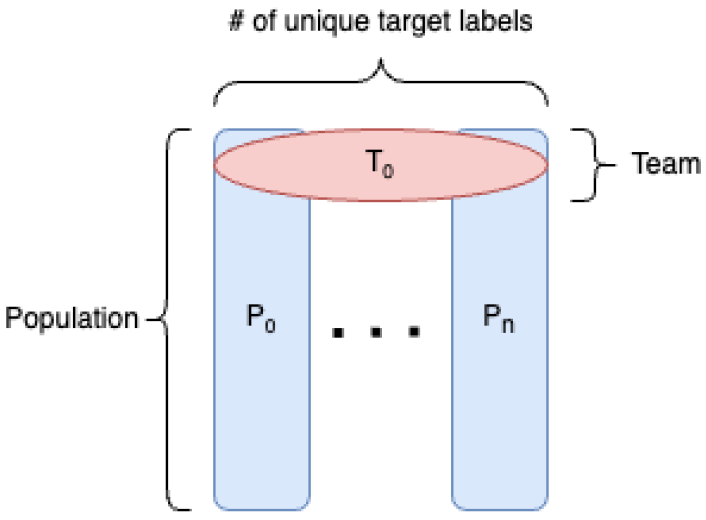

Figure 3.7: GP (LGP) Team Formation Diagram

Fitness Selection Criteria

The fitness selection in the (L)GP framework is a process that evaluates the predictive strength of teams [32], [33], not individuals. Each team's fitness score is determined by its ability to accurately predict dataset instances, and this score forms the basis of the evolutionary selection process.

Within a team, the evaluation begins at the individual level. Each individual's equation, representing its genotype, is applied to the dataset instances. Taking an example of an individual's equation such as $(F_0 + F_1) + (F_{40} - F_0) + (F_{23}/2)$, the feature values from an instance are substituted into the equation, and the result is computed. This process is repeated for all individuals in a team to ascertain their respective classification scores.

The individual within the team that yields the highest value for a given instance dictates the team's classification for that instance. The fitness score is dictated by comparing the predicted labels against the true classification of the instance, with correct classifications enhancing the team's overall fitness score.

Crossover and Mutation

Crossover is a genetic operation used to combine the information of two parent individuals to produce new offspring. In the context of my GP algorithm, I employ both single-point and double-point crossover methods with equal probability. For a given pair of parents within the same population, a single crossover point is chosen for single-point crossover, while two points are selected for double-point crossover. Parts of the parent equations are then exchanged at these points to create two new offspring, maintaining the diversity within a population that is evolving towards predicting a specific target label. The crossover process is denoted for a single point as:

$$\text{Offspring}_1 = \text{Parent}_1[: c] + \text{Parent}_2[c :] \tag{3.6}$$

$$\text{Offspring}_2 = \text{Parent}_2[: c] + \text{Parent}_1[c :] \tag{3.7}$$

and similarly extended for double-point crossover.

Mutation follows crossover and involves altering a single operation or operand within an offspring's equation. This process introduces new genetic variations and is parametrized to occur with a certain probability, implying that not all offspring from crossover will experience

mutation. The mutation can be represented as:

$$\text{Mutated}_i = \text{Offspring}_i[\text{operation}] \text{ or } \text{Offspring}_i[\text{operand}] \tag{3.8}$$

where the operation or operand is selected and changed randomly.

Newly created and possibly mutated offspring are then incorporated into their corresponding population. With each generation, the algorithm maintains the top n performing teams based on fitness scores and replace the remainder of the population with offspring generated through crossover and mutation. These fresh teams are then evaluated for fitness, contributing to the evolving population. This iterative cycle of selection, genetic operations, and fitness evaluation continues, driving the GP system towards optimal predictive performance.

Genetic Programming with Fitness Sharing

Fitness Sharing in (L)GP is a strategy employed to maintain diversity within the population of solutions, aiming to prevent premature convergence on sub-optimal solutions. It operates on the principle that individuals within a population should share their fitness with other similar individuals, effectively reducing the fitness of common solutions and encouraging the exploration of the solution space. The fitness sharing mechanism used in this study is based on the methodologies developed by Lichodzijewski et al. [33] in genetic programming.

The fitness sharing mechanism is represented as:

$$\frac{G(g_i, p_k)}{\sum G(g_j, p_k)} \tag{3.9}$$

where $G(g_i, p_k)$ denotes the fitness of a team g_i with respect to a data point p_k, and the denominator represents the sum of fitness scores of all teams g_j for the data point p_k. This formulation ensures that teams are rewarded not only for their predictive accuracy but also for their uniqueness in the population.

In the (L)GP implementation, teams evolve across generations to optimize the amount of instances predicted, which is utilized as the fitness function. The evolutionary process spans 1000 generations to identify the champion team, distinguished by selecting the highest amount of instances that is correctly predicted by each team. The algorithm, executed 20 times with and without Fitness Sharing, facilitates the derivation of the most effective reduced feature set for transforming the original training dataset.

Genetic Programming without Fitness Sharing

In contrast to the approach that employs fitness sharing, (L)GP without fitness sharing retains the raw fitness scores of teams. This conventional method evaluates teams based on their predictive accuracy without adjusting for the diversity of solutions within the population. The absence of fitness sharing means that each team's fitness is assessed in isolation from others, strictly based on its success in predicting dataset instances.

Without the mechanism to discourage the clustering of similar solutions, GP without fitness sharing may exhibit faster convergence, but with an increased risk of premature convergence to sub-optimal solutions. However, this direct approach allows for a straightforward selection process, where the fitness of a team is the sole criterion for its survival and reproduction. Teams with higher raw fitness scores are considered more fit and are more likely to be selected for generating offspring in the next generation.

The proposed (L)GP model applies this non-adjusted fitness evaluation as a baseline to contrast the effects of fitness sharing. By executing the GP algorithm in this manner, it's possible to observe the impact of diversity preservation on the evolutionary process and the resulting feature set effectiveness.

3.5 Sampling Techniques

This section explores the sampling techniques used for each dataset. I initially partition the data into stratified testing and training subsets, using an 80/20 split to ensure that the representation of classes within both subsets accurately reflects the overall dataset. Following this, I apply oversampling and undersampling techniques exclusively to the stratified training data. This methodology is utilized across both the original dataset and those obtained through dimensionality reduction techniques such as PCA, ICA, and GP.

3.5.1 Stratified Sampling

Stratified sampling is a technique where the dataset D is divided into K homogeneous subgroups or strata $\{S_1, S_2, \ldots, S_K\}$ based on the class labels. Samples are then drawn from each stratum to ensure proportional representation:

$$D_{train} \cup D_{test} = \bigcup_{k=1}^{K} S'_k \tag{3.10}$$

where S'_k is a random sample from stratum S_k. This preserves the original class distributions in both training D_{train} and testing sets D_{test}. For an extensive understanding and applications of this technique, please see [34].

3.5.2 Oversampling

Oversampling targets the imbalance in class distribution by augmenting the minority class C_{min} in the training set. If the minority class has n_{min} instances and the majority n_{max}, oversampling replicates the minority instances to satisfy:

$$|C'_{min}| = n_{max} \tag{3.11}$$

where C'_{min} is the augmented minority class. This results in a balanced training set, allowing for equal learning opportunities across classes. For an advanced approach to generating synthetic samples, please see [35].

3.5.3 Undersampling

In contrast, undersampling reduces the prevalence of the majority class C_{max} by randomly removing instances until the class sizes are balanced:

$$|C'_{max}| = n_{min} \tag{3.12}$$

where C'_{max} is the reduced majority class. This process leads to a smaller, balanced dataset $D_{balanced}$ that the model can learn from more equitably. For advanced techniques and methodologies on undersampling, please see [36].

3.6 Supervised Learning Techniques

In this section, machine learning models employed are summarized.

3.6.1 Decision Tree (DT)

A Decision Tree is a non-parametric supervised learning algorithm widely employed in both classification and regression tasks. It operates by recursively splitting the data set into increasingly specific subsets based on the most discriminative features. This splitting results in a tree-like model of decisions, where each internal node represents a decision based on a

single input feature, each branch represents the outcome of that decision, and each leaf node represents a class label or continuous outcome. For an in-depth discussion and practical examples of decision trees, please see [37].

The decision at each node is made using criteria such as entropy or Gini impurity, which help in selecting the feature that best separates the data into homogenous sets. Entropy, a measure of impurity or randomness, is calculated using the formula:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{3.13}$$

where p_i is the probability of an element being classified to a specific class. Alternatively, Gini impurity can be used and is defined as:

$$G(S) = 1 - \sum_{i=1}^{n} p_i^2 \tag{3.14}$$

The feature providing the highest information gain, calculated as the difference in entropy or Gini impurity before and after the split, is selected for splitting at each node:

$$IG(T, a) = H(T) - \sum_{v \in Values(a)} \frac{|T_v|}{|T|} H(T_v) \tag{3.15}$$

where T denotes the training set, a the attribute, and T_v the subset of T for a particular value v of attribute a.

3.6.2 Random Forest (RF)

The Random Forest algorithm is an advanced ensemble learning technique used for both classification and regression tasks. It enhances the stability and accuracy of classifications by combining the results of multiple Decision Trees, each constructed using a subset of the training data and a subset of the features. For an extensive discussion on Random Forests, including their implementation and optimization, please see [38].

RF addresses the overfitting problem common in single DTs by averaging multiple DTs that individually suffer from high variance but are decorrelated from each other. This approach not only improves classification accuracy but also helps in achieving robustness against noise present in the training data. The key elements of RF include:

- **Bootstrap Aggregating (Bagging):** Each tree is built on a bootstrap sample of the data — a random selection with replacement. This introduces variability among the trees.

- **Feature Randomness:** When growing each tree, at each split, a random subset of features is considered. This ensures that the trees are different and adds an extra layer of diversity to the model.

Mathematically, the classification of the RF is obtained by averaging the classifications from all the trees. This reduces the variance component of the error, as indicated by the equation:

$$V(\hat{Y}_{RF}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{3.16}$$

where $\hat{Y}_{RF}$ is the RF classification, ρ is the correlation between any two trees in the forest, σ^2 is the variance of the trees, and B is the number of trees.

3.6.3 Gaussian Naive Bayes (GNB)

The Gaussian Naive Bayes classifier is a probabilistic model that adapts the Naive Bayes approach for continuous data, assuming that the features associated with each class follow a Gaussian distribution. This makes it suitable for many real-world scenarios where data features are continuous. For a feature x_i in class y, the probability $P(x_i|y)$ is estimated using the Gaussian probability density function:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \tag{3.17}$$

where μ_y and σ_y^2 are the mean and variance of the features for class y, estimated from the data. For an extensive discussion on Gaussian Naive Bayes and its applications, please see [39].

The classifier's effectiveness hinges on the assumption of conditional independence among features given the class, simplifying the computation of probabilities. While this assumption can limit the classifier in cases of feature correlation, GNB is still efficient and performs well in scenarios like text classification and medical diagnosis, where the independence assumption is reasonably met.

Despite its simplicity, GNB can rival more complex models when the data's distribution aligns well with its assumptions. However, its performance might falter with highly correlated features. Techniques like feature selection can sometimes mitigate this issue. In essence, GNB provides a straightforward yet powerful approach to classification problems involving continuous data, balancing performance and computational efficiency effectively.

3.6.4 K-Nearest Neighbors (kNN)

The K-Nearest Neighbors algorithm is a non-parametric method used primarily for classification and regression. By design, kNN is based on a simple principle: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its nearest neighbors. For practical implementations and a deeper understanding of kNN, please see [40].

In the kNN approach, the input consists of the k closest training examples in the feature space. The output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors measured by a distance function. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

The kNN algorithm involves a simple but fundamental distance metric to measure the closeness of instances. The most commonly used metric is the Euclidean distance given by:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3.18}$$

where x and y are two points in a n-dimensional space and $d(x, y)$ is the Euclidean distance between them. The Euclidean distance effectively governs how the nearest neighbors are identified, influencing the classification or regression outcome.

The probability that an observation x belongs to the default class can be modeled as:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}} \tag{3.19}$$

where z is the linear combination of the input features x and the model parameters β.

3.6.5 Logistic Regression (LR)

Logistic Regression is a statistical method for binary classification. It extends the linear regression model by applying a logistic function to the output, enabling the model to estimate the probability of a binary outcome based on one or more predictor variables. For a thorough discussion on logistic regression and its applications across various fields, please see [41].

LR models the probability that a given input point belongs to a particular category. This is particularly useful in cases where the outcome to be predicted is categorical (usually binary: Yes/No, True/False). The logistic function, also known as the sigmoid function, is what transforms the linear equation into a probability measure that ranges between 0 and 1.

The probability that an observation x belongs to the default class (often represented as "1") can be modeled as:

$$P(y = 1|x) = \frac{1}{1 + e^{-z}} \tag{3.20}$$

where z is the linear combination of the input features x and the model parameters β, given by:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{3.21}$$

The coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are learned during model training through a process of maximizing the likelihood of the observed data, often using methods like gradient descent or other optimization algorithms.

3.6.6 XGBoost (XGB)

XGBoost, which stands for eXtreme Gradient Boosting, is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGB provides a scalable machine learning system for tree boosting that is widely used in winning solutions for many data science competitions. For a detailed exploration of XGBoost and its applications, please see [42].

XGB improves upon the traditional gradient boosting method by introducing a more regularized model formalization to control over-fitting, which gives it better performance. This is achieved through both hardware and algorithmic enhancements, including systems optimizations such as cache access patterns, data compression, and sharding.

At its core, XGB uses a series of DTs, where each tree is built in a sequential manner. Each new tree in the series attempts to correct the errors or residuals made by the previous trees. The final classification is an ensemble of these weak classification models. Mathematically, the classification model for a given data set D with n features is defined as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{3.22}$$

where K is the number of trees, f_k represents a tree, $\mathcal{F}$ is the space of all possible trees, and x_i is the vector of predictors for the ith instance.

XGB's learning objective combines a specific loss function to be optimized and a regularization term, which helps in reducing model complexity and overfitting. The general form of the objective function can be represented as:

$$\text{Obj}(\Theta) = L(\Theta) + \Omega(\Theta) \tag{3.23}$$

where L is the loss function that measures the difference between the predicted and actual values, and Ω represents the regularization term, which is typically the sum of the norms of the parameters of the model.

3.7 Summary

In this chapter, the methodology employed in this thesis has been outlined, covering the selection and analysis of datasets, feature extraction techniques, sampling strategies, and machine learning models utilized for infiltration attacks in insider threat detection. The datasets analyzed — CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2 — were meticulously preprocessed to ensure data integrity and relevance for the subsequent stages of analysis. For the CERT r4.2 dataset, a granular approach was adopted, analyzing the data at weekly, daily, and session-based granularity levels, which allowed for a detailed examination of user behavior over varying time frames.

Feature extraction techniques such as Genetic Programming with (GP-FS) and without (GP-NFS) Fitness Sharing, Principal Component Analysis (PCA) and Independent Component Analysis (ICA) were implemented to effectively reduce the dimensionality of the datasets. The aim was to enhance the computational efficiency of the machine learning

models. The application of diverse sampling techniques, including stratified sampling, oversampling, and undersampling, addressed potential imbalances in the datasets, facilitating a more equitable training environment.

The machine learning models deployed, including DT, RF, GNB, K-Nearest Neighbors, LR, and XGB, were evaluated for their efficacy in detecting insider threats. Each model was chosen based on its potential to provide distinct insights into the problem, thereby enriching the analysis.

The combination of these methodologies forms a comprehensive framework designed to explore the effectiveness of dimensionality reduction and data sampling in enhancing the performance of machine learning models in the context of infiltration attacks for insider threat detection.

Chapter 4

Evaluation and Results

This chapter discussed the results and observations of experiments performed for this thesis research.

4.1 Experimental Setup

In this research, experiments are performed on a machine with specifications including an Nvidia GeForce RTX 3070 GPU, 32GB of RAM, and an Intel Core i7-12700K processor. This ensured sufficient computational resources for the data processing and analysis tasks.

Dimensionality reduction is conducted using PCA, and ICA (namely Fast ICA) with scikit-learn's default parameters, producing feature sets of 5, 10, 15, and 20 components to evaluate the impact on model efficiency and performance. Moreover, the LGP is implemented with and without fitness sharing for dimensionality reduction, GP-FS and GP-NFS, respectively. To this end, the specific parameters used are: individual count set to 100, operation count set to 5, gap percentage set to 0.8, mutation probability set to 0.5, and generations were over 10,000. The population count matched the dataset's unique target labels, with resampling at every 100 generations, and a sample size of 10,000 instances.

As for the supervised machine learning models used, I utilize default settings for scikit-learn's supervised learning algorithms including DT, FR, GNB, kNN, LR, and XGB. The objective is to explore the effectiveness of these models trained on reduced feature sets derived from PCA, ICA, GP-FS, and GP-NFS, compared to models trained by using the complete feature set of the original datasets.

4.2 Performance Metrics

In order to evaluate the effectiveness of the models, several performance metrics are utilized: F1 Macro, Precision, and Recall. These metrics are crucial for understanding model behavior, especially in scenarios with imbalanced classes.

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is a measure of a classifier's exactness. A higher precision score indicates a model that yields more relevant results. Precision is particularly important in situations where the cost of a false positive is high. The formula for precision is given by:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

where TP represents true positives and FP represents false positives.

Recall

Recall, also known as sensitivity, measures the ability of a model to find all the relevant cases (i.e., true positives) within a dataset. High recall is crucial in cases where missing a positive instance is significantly detrimental. The formula for recall is:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

where TP represents true positives and FN represents false negatives.

F1-Score

The F1-score is the harmonic mean of precision and recall, providing a balance between them. It is particularly useful when the class distribution is uneven. F1-Score averages the F1 scores, calculated separately for each class, hence taking into account the balance between precision and recall across all classes without being influenced by any class imbalance. The formula for F1-Score is:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

This metric is especially critical in assessing the overall performance of the model across diverse scenarios and datasets.

4.3 CIC-IDS2017 Stratified Performance Results

In this section, results on the evaluations performed using the CIC-IDS2017 Stratified datasets are presented.

4.3.1 CIC-IDS2017 stratified results with Original features

ML Model	F1-Score	Precision	Recall
DT	**0.982**	0.982	0.983
RF	**0.971**	0.987	0.957
GNB	0.443	0.397	0.500
kNN	0.904	0.927	0.884
LR	0.443	0.397	0.500
XGB	**0.979**	0.989	0.970

Table 4.1: CIC-IDS2017 Stratified results for the Original

The analysis of the CIC-IDS2017 dataset with the original 77 features, shown in Table 4.1. DT, RF, and XGB are the top performers based on F1-Score and Precision. The DT leads with an F1-Score and Precision of 0.982, followed by RF with the highest Precision at 0.987 and an F1-Score of 0.971. XGB also demonstrates robust performance with a Precision of 0.989 and an F1-Score of 0.979. These models are effective for insider threat detection in cybersecurity, offering high accuracy and minimal false positives.

4.3.2 CIC-IDS2017 stratified results with PCA features

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.844	0.831	0.799	0.843
	Precision	0.843	0.824	0.785	0.835
	Recall	0.845	0.839	0.819	0.851
RF	F1-Score	0.889	0.890	0.873	**0.899**
	Precision	0.939	0.939	0.923	0.945
	Recall	0.856	0.856	0.840	0.867
Gaussian NB	F1-Score	0.524	0.524	0.579	0.731
	Precision	0.593	0.594	0.609	0.726
	Recall	0.639	0.641	0.666	0.737
k-NN	F1-Score	0.881	0.880	0.888	**0.897**
	Precision	0.912	0.907	0.916	0.922
	Recall	0.857	0.860	0.866	0.877
LR	F1-Score	0.803	0.808	0.807	0.811
	Precision	0.917	0.900	0.891	0.899
	Recall	0.755	0.763	0.765	0.767
XGB	F1-Score	0.849	0.855	0.856	**0.871**
	Precision	0.934	0.937	0.929	0.934
	Recall	0.804	0.811	0.814	0.833

Table 4.2: CIC-IDS2017 Stratified results for PCA

An analysis of the CIC-IDS2017 dataset using PCA-reduced features is detailed in Table 4.2. Among the PCA configurations, PCA-20 consistently shows the highest performance for top models when compared to other PCA settings. Specifically, RF with PCA-20 achieves an F1-Score of 0.899 and Precision of 0.945, approaching its performance with the original features (F1-Score of 0.971 and Precision of 0.987). XGB and k-NN also exhibit robust results at PCA-20, with XGB achieving an F1-Score of 0.871 and Precision of 0.934, and k-NN achieving an F1-Score of 0.897 and Precision of 0.922. These models retain substantial detection performance with PCA-20, offering a streamlined feature set while maintaining comparable performance to the original 77-feature dataset.

4.3.3 CIC-IDS2017 stratified results with ICA features

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.842	0.893	0.895	**0.912**
	Precision	0.855	0.894	0.894	0.910
	Recall	0.831	0.893	0.896	0.914
RF	F1-Score	0.898	0.926	0.928	**0.942**
	Precision	0.942	0.960	0.962	0.972
	Recall	0.867	0.900	0.901	0.918
Gaussian NB	F1-Score	0.314	0.404	0.395	0.514
	Precision	0.562	0.570	0.569	0.592
	Recall	0.546	0.581	0.576	0.636
k-NN	F1-Score	0.881	0.887	0.891	0.897
	Precision	0.912	0.917	0.919	0.921
	Recall	0.858	0.863	0.869	0.878
LR	F1-Score	0.443	0.808	0.443	0.811
	Precision	0.397	0.901	0.397	0.900
	Recall	0.500	0.764	0.500	0.767
XGB	F1-Score	0.857	0.884	0.891	**0.919**
	Precision	0.927	0.940	0.948	0.963
	Recall	0.816	0.848	0.854	0.887

Table 4.3: CIC-IDS2017 Stratified results for ICA

Table 4.3 shows the performance metrics of machine learning models using ICA-reduced features. ICA-20 outperforms other ICA configurations for the top models. Specifically, RF with ICA-20 reaches an F1-Score of 0.942 and Precision of 0.972, which surpasses its performance with PCA-20 (F1-Score of 0.899, Precision of 0.945) and closely approaches the performance with the original 77 features (F1-Score of 0.971, Precision of 0.987). XGB at

ICA-20 also excels, achieving an F1-Score of 0.919 and Precision of 0.963. DT with ICA-20 achieves an F1-Score of 0.912 and Precision of 0.910, which is an improvement over its PCA-20 performance (F1-Score of 0.843, Precision of 0.835). These results suggest that ICA-20, with only 20 features, provides an effective dimensionality reduction while maintaining or even enhancing model performance compared to both PCA-reduced and original feature sets.

4.3.4 CIC-IDS2017 stratified results with GP features

ML Model	Metric	GP-FS-13	GP-NFS-10
DT	F1-Score	0.813	0.816
	Precision	0.854	0.877
	Recall	0.786	0.781
RF	F1-Score	0.830	**0.817**
	Precision	0.900	0.881
	Recall	0.790	0.781
Gaussian NB	F1-Score	0.443	0.322
	Precision	0.397	0.564
	Recall	0.500	0.549
k-NN	F1-Score	0.826	**0.826**
	Precision	0.901	0.911
	Recall	0.786	0.782
LR	F1-Score	0.443	0.793
	Precision	0.397	0.854
	Recall	0.500	0.760
XGB	F1-Score	0.832	**0.827**
	Precision	0.945	0.941
	Recall	0.780	0.776

Table 4.4: CIC-IDS2017 Stratified results for GP-FS-13 and GP-NFS-10

Table 4.4 examines the impact of Genetic Programming (GP) feature reduction, both with Fitness Sharing (GP-FS-13) and without (GP-NFS-10), on machine learning model performance, using 10 and 13 features. Among the models, XGB with GP-FS-13 achieves an F1-Score of 0.832 and Precision of 0.945, showing a notable performance considering the reduced feature count, though slightly lower than its performance with ICA-20 (F1-Score of 0.919, Precision of 0.963) and the original features (F1-Score of 0.979, Precision of 0.989). RF also demonstrates competitive results with GP-FS-13 (F1-Score of 0.830, Precision of 0.900) compared to its PCA-20 and original performances. These results highlight that GP, with fewer than 15 features, manages to retain a high portion of the models' detection

capabilities, offering a promising avenue for feature reduction in cybersecurity applications.

4.4 CSE-CIC-IDS2018 Stratified Performance Results

In this section, results on the evaluations performed using the CSE-CIC-IDS2018 Stratified datasets are presented.

4.4.1 CSE-CIC-IDS2018 stratified results with Original features

ML Model	F1-Score	Precision	Recall
DT	0.999	0.999	**0.999**
RF	0.999	0.999	**0.999**
Gaussian NB	0.183	0.602	0.507
k-NN	0.979	0.987	0.971
LR	0.805	0.789	0.827
XGB	0.999	1.000	**0.999**

Table 4.5: CSE-CIC-IDS2018 Stratified results for the Original

The analysis of the CSE-CIC-IDS2018 dataset with the original 77 features, shown in Table 4.5, identifies DT, RF, and XGB as top performers based on F1-Score and Precision. DT and RF both achieve an F1-Score and Precision of 0.999. XGB also demonstrates strong performance with an F1-Score of 0.999 and a Precision of 1.000. These models exhibit high accuracy in detecting threats within the dataset.

4.4.2 CSE-CIC-IDS2018 stratified results with PCA features

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.984	0.957	0.956	0.959
	Precision	0.984	0.974	0.972	0.976
	Recall	0.984	0.942	0.941	0.944
RF	F1-Score	0.991	0.993	0.994	**0.995**
	Precision	0.995	0.997	0.997	0.997
	Recall	0.987	0.990	0.991	0.992
Gaussian NB	F1-Score	0.945	0.948	0.948	0.949
	Precision	0.952	0.957	0.958	0.958
	Recall	0.938	0.940	0.939	0.940
k-NN	F1-Score	0.990	0.993	0.994	**0.994**
	Precision	0.993	0.995	0.996	0.996
	Recall	0.986	0.991	0.992	0.993
LR	F1-Score	0.950	0.950	0.953	0.958
	Precision	0.961	0.961	0.964	0.970
	Recall	0.940	0.941	0.942	0.946
XGB	F1-Score	0.988	0.991	0.993	**0.994**
	Precision	0.993	0.995	0.996	0.997
	Recall	0.983	0.987	0.990	0.992

Table 4.6: CSE-CIC-IDS2018 Stratified results for PCA

Table 4.6 presents the performance of machine learning models on the CSE-CIC-IDS2018 dataset using PCA-reduced features. For the top-performing models, PCA-20 emerges as the most effective configuration. RF, with PCA-20, attains an F1-Score of 0.995 and Precision of 0.997, which is nearly identical to its performance with the original 77 features (F1-Score of 0.999 and Precision of 0.999). XGB also performs well with PCA-20, achieving an F1-Score of 0.994 and Precision of 0.997, closely mirroring its original feature performance. Similarly, k-NN shows strong results with PCA-20, with an F1-Score of 0.994 and Precision of 0.996.

4.4.3 CSE-CIC-IDS2018 stratified results with ICA features

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.977	0.991	0.992	**0.993**
	Precision	0.985	0.992	0.992	0.993
	Recall	0.971	0.991	0.992	0.993
RF	F1-Score	0.992	0.995	0.995	**0.996**
	Precision	0.995	0.998	0.998	0.998
	Recall	0.990	0.992	0.993	0.994
Gaussian NB	F1-Score	0.835	0.839	0.841	0.838
	Precision	0.806	0.810	0.813	0.810
	Recall	0.895	0.894	0.895	0.888
k-NN	F1-Score	0.984	0.993	0.992	0.993
	Precision	0.980	0.996	0.991	0.991
	Recall	0.988	0.991	0.994	0.994
LR	F1-Score	0.936	0.950	0.952	0.953
	Precision	0.933	0.961	0.963	0.963
	Recall	0.939	0.941	0.943	0.943
XGB	F1-Score	0.988	0.994	0.994	**0.995**
	Precision	0.988	0.997	0.997	0.997
	Recall	0.988	0.992	0.994	0.994

Table 4.7: CSE-CIC-IDS2018 Stratified results for ICA

Table 4.7 showcases the performance of machine learning models using ICA-reduced features on the CSE-CIC-IDS2018 dataset. Among the ICA configurations, ICA-20 shows the best performance for top models. RF with ICA-20 achieves an F1-Score of 0.996 and Precision of 0.998, closely matching its results with PCA-20 (F1-Score of 0.995 and Precision of 0.997) and the original 77 features (F1-Score of 0.999 and Precision of 0.999). XGB and k-NN also deliver strong results with ICA-20, with XGB achieving an F1-Score of 0.995 and Precision of 0.997, and k-NN achieving an F1-Score of 0.993 and Precision of 0.991.

4.4.4 CSE-CIC-IDS2018 stratified results with GP features

ML Model	Metric	GP-FS-15	GP-NFS-15
DT	F1-Score	0.985	**0.988**
	Precision	0.986	0.988
	Recall	0.984	0.988
RF	F1-Score	0.988	**0.990**
	Precision	0.992	0.992
	Recall	0.985	0.988
Gaussian NB	F1-Score	0.289	0.457
	Precision	0.608	0.635
	Recall	0.559	0.660
k-NN	F1-Score	0.987	**0.989**
	Precision	0.991	0.993
	Recall	0.984	0.986
LR	F1-Score	0.929	0.471
	Precision	0.957	0.888
	Recall	0.907	0.513
XGB	F1-Score	0.985	0.987
	Precision	0.990	0.992
	Recall	0.980	0.982

Table 4.8: CSE-CIC-IDS2018 Stratified results for GP-FS-15 and GP-NFS-15

Table 4.8 presents the performance of machine learning models using Genetic Programming (GP) with and without Fitness Sharing (FS and NFS) on the CSE-CIC-IDS2018 dataset. For top-performing models, GP-FS-15 and GP-NFS-15 both show strong results. RF with GP-NFS-15 achieves an F1-Score of 0.990 and Precision of 0.992, closely matching its performance with ICA-20 (F1-Score of 0.996 and Precision of 0.998), PCA-20 (F1-Score of 0.995 and Precision of 0.997), and the original features (F1-Score of 0.999 and Precision of 0.999). XGB and k-NN also perform well with GP-NFS-15, with XGB achieving an F1-Score of 0.987 and Precision of 0.992, and k-NN achieving an F1-Score of 0.989 and Precision of 0.993. These results indicate that GP, using 15 features, maintains high detection performance, providing a competitive alternative to both the original and other dimensionality reduction techniques.

4.5 CERT r4.2 Stratified Performance Results

In this section, results on the evaluations performed using the CERT r4.2 Stratified datasets are presented at different granularity levels.

4.5.1 CERT r4.2 Week

For the CERT r4.2 Week datasets, "0vsAll" is used hereafter to denote benign vs. all insider threat scenarios in the dataset. While, "0vs1_3" is used hereafter to denote benign vs. data exfiltration (scenario-1) and IT sabotage (scenario-3) scenarios.

CERT r4.2 Week Original Stratified

ML Model	0vsAll			0vs1_3		
	F1-Score	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**
DT	0.770	0.764	0.777	**1.000**	1.000	1.000
RF	0.731	0.998	0.651	**1.000**	1.000	1.000
Gaussian NB	0.499	0.498	0.500	0.500	0.500	0.500
k-NN	0.499	0.498	0.500	0.500	0.500	0.500
LR	0.499	0.498	0.500	0.500	0.500	0.500
XGB	0.872	0.935	0.825	**1.000**	1.000	1.000

Table 4.9: CERT r4.2 Week Stratified results for the Original features

The analysis of the CERT r4.2 dataset with week granularity level using the original 667 features, shown in Table 4.9, identifies XGB, DT, and RF as the top performers based on F1-Score and Precision. For the 0vsAll classification, XGB achieves the highest F1-Score of 0.872 and Precision of 0.935. DT follows with an F1-Score of 0.770 and Precision of 0.764, while RF shows a strong Precision of 0.998 but a lower F1-Score of 0.731. In the 0vs1_3 classification, all three models—XGB, DT, and RF—achieve very high scores with an F1-Score and Precision of 0.999.

CERT r4.2 Week PCA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.581	0.661	0.638	0.682	0.816	1.000	1.000	**1.000**
	Precision	0.586	0.674	0.642	0.670	0.928	1.000	1.000	1.000
	Recall	0.578	0.649	0.633	0.697	0.750	1.000	1.000	1.000
RF	F1-Score	0.586	0.659	0.681	0.691	0.868	1.000	0.978	**0.978**
	Precision	0.998	0.998	0.998	0.998	0.999	1.000	1.000	1.000
	Recall	0.548	0.595	0.611	0.619	0.792	1.000	0.958	0.958
Gaussian NB	F1-Score	0.510	0.527	0.521	0.515	0.514	0.546	0.537	0.528
	Precision	0.508	0.518	0.515	0.512	0.510	0.526	0.522	0.518
	Recall	0.537	0.584	0.582	0.593	0.739	0.992	0.990	0.988
k-NN	F1-Score	0.596	0.712	0.679	0.672	0.929	0.978	0.978	0.978
	Precision	0.887	0.998	0.965	0.887	1.000	1.000	1.000	1.000
	Recall	0.555	0.635	0.611	0.611	0.875	1.000	0.958	0.958
LR	F1-Score	0.499	0.648	0.659	0.659	0.571	1.000	1.000	**1.000**
	Precision	0.498	0.998	0.998	0.998	0.750	1.000	1.000	1.000
	Recall	0.500	0.587	0.595	0.595	0.542	1.000	1.000	1.000
XGB	F1-Score	0.606	0.719	0.713	0.725	0.850	0.955	0.954	0.954
	Precision	0.831	0.972	0.927	0.951	0.937	0.999	0.999	0.999
	Recall	0.563	0.643	0.643	0.651	0.792	0.917	0.917	0.917

Table 4.10: CERT r4.2 Week Stratified results for PCA

Table 4.10 presents the performance of machine learning models on the CERT r4.2 Week dataset using PCA-reduced features. Among the PCA configurations, PCA-20 consistently shows the highest performance. XGB with PCA-20 achieves an F1-Score of 0.725 and Precision of 0.951 for the 0vsAll classification, compared to its original feature performance (F1-Score of 0.872 and Precision of 0.935). For the 0vs1_3 classification, XGB maintains high scores with an F1-Score of 0.954 and Precision of 0.999, slightly lower than the scores with the original features. RF and DT also perform well with PCA-20, with RF achieving an F1-Score of 0.691 and Precision of 0.998 for 0vsAll, and both models scoring 1.000 for 0vs1_3. This shows the comparable performance of PCA-20 with reduced feature dimensionality while maintaining the detection performance against the original 667 features.

CERT r4.2 Week ICA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.607	0.655	0.662	0.623	0.958	1.000	1.000	**1.000**
	Precision	0.591	0.646	0.653	0.615	0.958	1.000	1.000	1.000
	Recall	0.632	0.665	0.673	0.633	0.958	1.000	1.000	1.000
RF	F1-Score	0.636	0.691	0.699	0.681	0.929	1.000	0.978	0.955
	Precision	0.998	0.998	0.969	0.998	1.000	1.000	1.000	1.000
	Recall	0.579	0.619	0.627	0.611	0.875	1.000	0.958	0.917
Gaussian NB	F1-Score	0.506	0.538	0.527	0.533	0.540	0.561	0.549	0.553
	Precision	0.506	0.525	0.518	0.522	0.523	0.534	0.528	0.530
	Recall	0.528	0.595	0.585	0.601	0.745	0.993	0.992	0.993
k-NN	F1-Score	0.591	0.699	0.689	0.697	0.929	0.978	0.978	**1.000**
	Precision	0.767	0.969	0.967	0.943	1.000	1.000	1.000	1.000
	Recall	0.555	0.627	0.619	0.627	0.875	0.958	0.958	1.000
LR	F1-Score	0.499	0.648	0.648	0.659	0.571	1.000	1.000	**1.000**
	Precision	0.498	0.998	0.998	0.998	0.750	1.000	1.000	1.000
	Recall	0.500	0.587	0.587	0.595	0.542	1.000	1.000	1.000
XGB	F1-Score	0.660	0.716	0.704	0.749	0.868	0.958	0.929	0.955
	Precision	0.790	0.948	0.923	0.938	1.000	0.958	1.000	1.000
	Recall	0.611	0.643	0.635	0.674	0.792	0.958	0.875	0.917

Table 4.11: CERT r4.2 Week Stratified results for ICA

Table 4.11 provides the performance metrics for machine learning models on the CERT r4.2 Week dataset using ICA-reduced features. Among the ICA configurations, ICA-20 demonstrates the highest performance for the top models. XGB with ICA-20 achieves an F1-Score of 0.749 and Precision of 0.938 for the 0vsAll classification, which is slightly lower than its original feature performance (F1-Score of 0.872 and Precision of 0.935). For the 0vs1_3 classification, XGB maintains high scores with an F1-Score of 0.955 and Precision of 1.000, closely matching the original feature performance. RF and DT also perform well with ICA-20, with RF achieving an F1-Score of 0.681 and Precision of 0.998 for 0vsAll, and DT achieving scores of 1.000 for 0vs1_3. These results suggest that ICA-20 can effectively reduce the feature count while maintaining a high level of performance comparable to the original 667 features.

CERT r4.2 Week GP Stratified

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS-13	**GP-NFS-14**	**GP-FS-14**	**GP-NFS-14**
DT	F1-Score	0.650	0.662	1.000	0.929
	Precision	0.781	0.706	1.000	1.000
	Recall	0.603	0.634	1.000	0.875
RF	F1-Score	0.647	0.632	0.955	**0.929**
	Precision	0.758	0.720	0.999	1.000
	Recall	0.603	0.595	0.917	0.875
Gaussian NB	F1-Score	0.544	0.540	0.643	0.500
	Precision	0.531	0.527	1.000	0.500
	Recall	0.589	0.581	0.583	0.500
k-NN	F1-Score	0.666	0.619	1.000	0.929
	Precision	0.931	0.873	1.000	1.000
	Recall	0.603	0.571	1.000	0.875
LR	F1-Score	0.648	0.499	1.000	**1.000**
	Precision	0.998	0.498	1.000	1.000
	Recall	0.587	0.500	1.000	1.000
XGB	F1-Score	0.664	0.634	0.955	**0.929**
	Precision	0.904	0.953	1.000	1.000
	Recall	0.603	0.579	0.917	0.875

Table 4.12: CERT r4.2 Week Stratified results for GP

Table 4.12 shows the performance of machine learning models on the CERT r4.2 Week dataset using Genetic Programming (GP) with and without Fitness Sharing (FS and NFS). Among the models, XGB with GP-FS-13 achieves an F1-Score of 0.664 and Precision of 0.904 for the 0vsAll classification, which is lower compared to its performance with the original 667 features (F1-Score of 0.872 and Precision of 0.935). For the 0vs1_3 classification, XGB achieves an F1-Score of 0.955 and Precision of 1.000, closely matching the results with the original features. RF and DT also perform well with GP-FS-13 and GP-NFS-14. RF achieves an F1-Score of 0.647 and Precision of 0.758 for 0vsAll, and an F1-Score of 0.955 and Precision of 0.999 for 0vs1_3. DT shows an F1-Score of 0.650 and Precision of 0.781 for 0vsAll, and an F1-Score of 1.000 and Precision of 1.000 for 0vs1_3. These results suggest that GP, using 13 and 14 features, can maintain a high level of performance comparable to that achieved with the original, more extensive feature set.

4.5.2 CERT r4.2 Day

For the CERT r4.2 Day datasets, "0vsAll" is used hereafter to denote benign vs. all insider threat scenarios in the dataset. While, "0vs1_3" is used hereafter to denote benign vs. data exfiltration (scenario-1) and IT sabotage (scenario-3) scenarios.

CERT r4.2 Day Original Stratified

ML Model	0vsAll			0vs1_3		
	F1-Score	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**
DT	0.899	0.920	0.881	**0.932**	0.913	0.952
RF	0.836	0.999	0.754	**0.962**	1.000	0.929
Gaussian NB	0.503	0.505	0.502	0.499	0.501	0.520
k-NN	0.499	0.499	0.500	0.500	0.500	0.500
LR	0.499	0.499	0.500	0.500	0.500	0.500
XGB	0.974	1.000	0.951	**0.962**	1.000	0.929

Table 4.13: CERT r4.2 Day Stratified results for the Original features

The analysis of the CERT r4.2 Day dataset with the original 507 features, shown in Table 4.13, identifies XGB, DT, and RF as the top performers based on F1-Score and Precision. For the 0vsAll classification, XGB achieves the highest F1-Score of 0.974 and Precision of 1.000. DT follows with an F1-Score of 0.899 and Precision of 0.920, while RF shows a strong Precision of 0.999 but a lower F1-Score of 0.836. In the 0vs1_3 classification, XGB and RF both achieve high scores, with XGB attaining an F1-Score of 0.962 and Precision of 1.000, and RF achieving an F1-Score of 0.962 and Precision of 1.000. DT also performs well with an F1-Score of 0.932 and Precision of 0.913. These results are based on models trained using the original feature set of 507 features.

CERT r4.2 Day PCA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.561	0.588	0.591	0.602	0.867	0.890	0.913	0.963
	Precision	0.557	0.585	0.585	0.592	0.821	0.900	0.880	0.975
	Recall	0.566	0.592	0.597	0.613	0.928	0.881	0.952	0.952
RF	F1-Score	0.570	0.582	0.583	0.579	0.947	0.962	0.962	**0.975**
	Precision	0.874	0.874	0.908	0.924	1.000	1.000	1.000	1.000
	Recall	0.539	0.547	0.547	0.544	0.905	0.929	0.929	0.952
Gaussian NB	F1-Score	0.504	0.506	0.503	0.501	0.511	0.520	0.509	0.506
	Precision	0.504	0.505	0.504	0.504	0.507	0.512	0.507	0.506
	Recall	0.519	0.528	0.530	0.539	0.756	0.970	0.966	0.964
k-NN	F1-Score	0.574	0.584	0.598	0.597	0.975	0.975	0.975	**0.975**
	Precision	0.880	0.805	0.854	0.832	1.000	1.000	1.000	1.000
	Recall	0.541	0.549	0.557	0.557	0.952	0.952	0.952	0.952
LR	F1-Score	0.515	0.549	0.548	0.548	0.728	0.950	0.950	0.939
	Precision	0.874	0.999	0.953	0.915	0.786	0.974	0.974	0.950
	Recall	0.508	0.526	0.526	0.526	0.690	0.929	0.929	0.929
XGB	F1-Score	0.581	0.607	0.618	0.662	0.936	0.962	0.962	**0.975**
	Precision	0.832	0.764	0.824	0.876	0.972	1.000	1.000	1.000
	Recall	0.547	0.567	0.572	0.604	0.905	0.929	0.929	0.952

Table 4.14: CERT r4.2 Day Stratified results for PCA

Table 4.14 presents the performance of machine learning models on the CERT r4.2 Day dataset using PCA-reduced features. Among the PCA configurations, PCA-20 demonstrates the highest performance for the top models. XGB with PCA-20 achieves an F1-Score of 0.662 and Precision of 0.876 for the 0vsAll classification, which is lower compared to its performance with the original 507 features (F1-Score of 0.974 and Precision of 1.000). For the 0vs1_3 classification, XGB achieves an F1-Score of 0.975 and Precision of 1.000, closely matching the results with the original features.

RF and k-NN also show strong results with PCA-20. RF achieves an F1-Score of 0.579 and Precision of 0.924 for 0vsAll, and an F1-Score of 0.975 and Precision of 1.000 for 0vs1_3. k-NN attains an F1-Score of 0.597 and Precision of 0.832 for 0vsAll, and an F1-Score of 0.975 and Precision of 1.000 for 0vs1_3. These results suggest that PCA-20 can reduce feature dimensionality while maintaining a substantial level of performance comparable to the original 507 features.

CERT r4.2 Day ICA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.583	0.610	0.640	0.616	0.872	0.880	0.913	0.909
	Precision	0.579	0.603	0.632	0.612	0.864	0.828	0.880	0.891
	Recall	0.587	0.618	0.649	0.621	0.881	0.952	0.952	0.929
RF	F1-Score	0.590	0.605	0.601	0.567	0.962	0.963	0.962	**0.975**
	Precision	0.869	0.959	0.977	0.999	1.000	0.975	1.000	1.000
	Recall	0.552	0.560	0.557	0.536	0.929	0.929	0.929	0.952
Gaussian NB	F1-Score	0.503	0.508	0.502	0.503	0.519	0.520	0.518	0.515
	Precision	0.503	0.506	0.504	0.504	0.511	0.512	0.511	0.510
	Recall	0.514	0.531	0.532	0.530	0.758	0.970	0.970	0.968
k-NN	F1-Score	0.572	0.576	0.581	0.597	0.975	0.962	0.975	**0.975**
	Precision	0.806	0.782	0.820	0.843	1.000	1.000	1.000	1.000
	Recall	0.541	0.544	0.547	0.557	0.952	0.929	0.952	0.952
LR	F1-Score	0.499	0.549	0.549	0.548	0.693	0.950	0.950	0.950
	Precision	0.499	0.999	0.999	0.883	0.800	0.974	0.974	0.974
	Recall	0.500	0.526	0.526	0.526	0.643	0.929	0.929	0.929
XGB	F1-Score	0.592	0.648	0.671	0.637	0.932	0.932	0.962	**0.975**
	Precision	0.817	0.915	0.870	0.850	1.000	1.000	1.000	1.000
	Recall	0.554	0.591	0.611	0.585	0.881	0.929	0.929	0.952

Table 4.15: CERT r4.2 Day Stratified results for ICA

Table 4.15 shows the performance of machine learning models on the CERT r4.2 Day dataset using ICA-reduced features. Among the ICA configurations, ICA-20 generally exhibits the highest performance. XGB with ICA-20 achieves an F1-Score of 0.637 and Precision of 0.850 for the 0vsAll classification, which is lower compared to its performance with the original 507 features (F1-Score of 0.974 and Precision of 1.000) and slightly lower than with PCA-20 (F1-Score of 0.662 and Precision of 0.876). For the 0vs1_3 classification, XGB achieves an F1-Score of 0.975 and Precision of 1.000, which is comparable to the results with both the original features and PCA-20.

RF and DT also perform well with ICA-20. RF attains an F1-Score of 0.567 and Precision of 0.999 for 0vsAll, and an F1-Score of 0.975 and Precision of 1.000 for 0vs1_3, which is comparable to its performance with PCA-20. DT achieves an F1-Score of 0.616 and Precision of 0.612 for 0vsAll, and an F1-Score of 0.909 and Precision of 0.891 for 0vs1_3.

CERT r4.2 Day GP Stratified

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS-12	**GP-NFS-12**	**GP-FS-12**	**GP-NFS-14**
DT	F1-Score	0.508	0.701	0.947	0.915
	Precision	0.528	0.769	1.000	0.925
	Recall	0.505	0.660	0.905	0.905
RF	F1-Score	0.504	0.694	0.905	**0.925**
	Precision	0.526	0.753	0.969	0.947
	Recall	0.502	0.658	0.857	0.905
Gaussian NB	F1-Score	0.515	0.500	0.500	0.750
	Precision	0.515	0.505	0.500	0.673
	Recall	0.517	0.563	0.500	0.952
k-NN	F1-Score	0.504	0.595	0.917	**0.947**
	Precision	0.599	0.738	1.000	1.000
	Recall	0.503	0.559	0.857	0.905
LR	F1-Score	0.499	0.534	0.932	**0.917**
	Precision	0.499	0.999	1.000	1.000
	Recall	0.500	0.518	0.881	0.857
XGB	F1-Score	0.504	0.539	0.917	0.895
	Precision	0.749	0.899	1.000	0.941
	Recall	0.503	0.521	0.857	0.857

Table 4.16: CERT r4.2 Day Stratified results for GP

Table 4.16 presents the performance of machine learning models on the CERT r4.2 Day dataset using Genetic Programming (GP) with and without Fitness Sharing (FS and NFS). GP for the Day granularity reduces the feature count to 12 and 14 features. Among the models, XGB with GP-FS-12 achieves an F1-Score of 0.917 and Precision of 1.000 for the 0vs1_3 classification, closely matching its performance with the original 507 features (F1-Score of 0.962 and Precision of 1.000). For the 0vsAll classification, XGB shows an F1-Score of 0.504 and Precision of 0.749 with GP-FS-12, lower than its original performance (F1-Score of 0.974 and Precision of 1.000).

RF with GP-NFS-12 attains an F1-Score of 0.694 and Precision of 0.753 for 0vsAll, and an F1-Score of 0.925 and Precision of 0.947 for 0vs1_3, demonstrating robust performance comparable to the original feature set. DT with GP-NFS-12 achieves an F1-Score of 0.701 and Precision of 0.769 for 0vsAll, and an F1-Score of 0.915 and Precision of 0.925 for 0vs1_3.

4.5.3 CERT r4.2 Session

For the CERT r4.2 Session datasets, "0vsAll" is used hereafter to denote benign vs. all insider threat scenarios in the dataset. While, "0vs1_3" is used hereafter to denote benign vs. data exfiltration (scenario-1) and IT sabotage (scenario-3) scenarios.

CERT r4.2 Session Original Stratified

ML Model	0vsAll			0vs1_3		
	F1-Score	**Precision**	**Recall**	**F1-Score**	**Precision**	**Recall**
DT	0.871	0.881	0.861	**0.987**	1.000	0.975
RF	0.802	0.994	0.717	**0.894**	1.000	0.825
Gaussian NB	0.499	0.499	0.500	0.500	0.500	0.500
k-NN	0.499	0.499	0.500	0.500	0.500	0.500
LR	0.499	0.499	0.500	0.500	0.500	0.500
XGB	0.961	0.985	0.939	**0.987**	1.000	0.975

Table 4.17: CERT r4.2 Session Stratified results for the Original features

The analysis of the CERT r4.2 Session dataset with the original 127 features, shown in Table 4.17, identifies XGB, DT, and RF as the top performers based on F1-Score and Precision. For the 0vsAll classification, XGB achieves the highest F1-Score of 0.961 and Precision of 0.985. DT follows with an F1-Score of 0.871 and Precision of 0.881, while RF shows a strong Precision of 0.994 but a lower F1-Score of 0.802. In the 0vs1_3 classification, both XGB and DT achieve high scores, with XGB attaining an F1-Score of 0.987 and Precision of 1.000, and DT also achieving an F1-Score of 0.987 and Precision of 1.000. RF achieves an F1-Score of 0.894 and Precision of 1.000. These results are based on models trained using the original feature set of 127 features.

CERT r4.2 Session PCA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.559	0.592	0.590	0.596	0.929	0.925	0.915	**0.959**
	Precision	0.557	0.587	0.588	0.598	1.000	0.925	0.905	1.000
	Recall	0.562	0.598	0.593	0.593	0.875	0.925	0.925	0.925
RF	F1-Score	0.558	0.566	0.570	0.566	0.929	0.929	0.929	**0.959**
	Precision	0.966	0.969	0.946	0.999	1.000	1.000	1.000	1.000
	Recall	0.531	0.536	0.538	0.536	0.875	0.875	0.875	0.925
Gaussian NB	F1-Score	0.525	0.510	0.503	0.500	0.789	0.521	0.524	0.531
	Precision	0.536	0.507	0.503	0.503	0.760	0.512	0.513	0.517
	Recall	0.520	0.521	0.520	0.528	0.825	0.822	0.822	0.823
k-NN	F1-Score	0.560	0.616	0.620	0.645	0.894	0.894	0.894	0.894
	Precision	0.811	0.749	0.787	0.817	1.000	1.000	1.000	1.000
	Recall	0.534	0.576	0.576	0.594	0.825	0.825	0.825	0.825
LR	F1-Score	0.538	0.538	0.538	0.538	0.894	0.894	0.894	0.894
	Precision	0.999	0.999	0.999	0.999	1.000	1.000	1.000	1.000
	Recall	0.520	0.520	0.520	0.520	0.825	0.825	0.825	0.825
XGB	F1-Score	0.549	0.585	0.598	0.645	0.912	0.912	0.912	**0.929**
	Precision	0.799	0.822	0.824	0.884	1.000	1.000	1.000	1.000
	Recall	0.527	0.549	0.558	0.590	0.850	0.850	0.850	0.875

Table 4.18: CERT r4.2 Session Stratified results for PCA

Table 4.18 shows the performance of machine learning models on the CERT r4.2 Session dataset using PCA-reduced features. Among the PCA configurations, PCA-20 demonstrates the highest performance. For the 0vsAll classification, XGB with PCA-20 achieves an F1-Score of 0.645 and Precision of 0.884, compared to its original feature performance (F1-Score of 0.961 and Precision of 0.985). k-NN with PCA-20 achieves an F1-Score of 0.645 and Precision of 0.817, and DT with PCA-20 achieves an F1-Score of 0.596 and Precision of 0.598.

For the 0vs1_3 classification, XGB, k-NN, and DT maintain high scores with PCA-20. XGB achieves an F1-Score of 0.929 and Precision of 1.000, k-NN achieves an F1-Score of 0.894 and Precision of 1.000, and DT achieves an F1-Score of 0.959 and Precision of 1.000.

CERT r4.2 Session ICA Stratified

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.591	0.620	0.634	0.647	0.878	0.925	0.962	**0.917**
	Precision	0.589	0.616	0.630	0.637	0.840	0.925	0.974	0.969
	Recall	0.593	0.625	0.638	0.658	0.925	0.925	0.950	0.875
RF	F1-Score	0.570	0.591	0.589	0.596	0.885	0.929	0.929	**0.894**
	Precision	0.946	0.896	0.999	0.979	0.895	1.000	1.000	1.000
	Recall	0.538	0.552	0.550	0.554	0.875	0.875	0.875	0.825
Gaussian NB	F1-Score	0.515	0.508	0.503	0.499	0.537	0.515	0.534	0.530
	Precision	0.512	0.505	0.503	0.502	0.520	0.509	0.519	0.516
	Recall	0.518	0.531	0.532	0.518	0.823	0.821	0.823	0.823
k-NN	F1-Score	0.552	0.573	0.622	0.652	0.894	0.894	0.894	0.894
	Precision	0.794	0.707	0.814	0.838	1.000	1.000	1.000	1.000
	Recall	0.529	0.545	0.576	0.599	0.825	0.825	0.825	0.825
LR	F1-Score	0.499	0.538	0.538	0.538	0.875	0.894	0.894	0.894
	Precision	0.499	0.999	0.999	0.883	1.000	1.000	1.000	1.000
	Recall	0.500	0.520	0.520	0.520	0.800	0.825	0.825	0.825
XGB	F1-Score	0.554	0.624	0.648	0.726	0.917	0.912	0.929	**0.974**
	Precision	0.881	0.892	0.886	0.920	0.969	1.000	1.000	1.000
	Recall	0.529	0.573	0.592	0.655	0.875	0.850	0.875	0.950

Table 4.19: CERT r4.2 Session Stratified results for ICA

Table 4.19 shows the performance of machine learning models on the CERT r4.2 Session dataset using ICA-reduced features. Among the ICA configurations, ICA-20 demonstrates the highest performance. For the 0vsAll classification, XGB with ICA-20 achieves an F1-Score of 0.726 and Precision of 0.920, compared to its original feature performance (F1-Score of 0.961 and Precision of 0.985). DT with ICA-20 achieves an F1-Score of 0.647 and Precision of 0.637, and k-NN with ICA-20 achieves an F1-Score of 0.652 and Precision of 0.838.

For the 0vs1_3 classification, XGB, DT, and k-NN maintain high scores with ICA-20. XGB achieves an F1-Score of 0.974 and Precision of 1.000, DT achieves an F1-Score of 0.917 and Precision of 0.969, and k-NN achieves an F1-Score of 0.894 and Precision of 1.000. These results suggest that ICA-20 effectively reduces the feature count while retaining a substantial level of performance, comparable to the original and PCA-reduced feature sets.

CERT r4.2 Session GP Stratified

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS-8	**GP-NFS-10**	**GP-FS-11**	**GP-NFS-13**
DT	F1-Score	0.514	0.655	0.894	0.833
	Precision	0.520	0.733	1.000	0.875
	Recall	0.511	0.616	0.825	0.800
RF	F1-Score	0.499	0.653	0.894	**0.853**
	Precision	0.499	0.716	1.000	0.929
	Recall	0.500	0.618	0.825	0.800
Gaussian NB	F1-Score	0.513	0.509	0.522	0.561
	Precision	0.511	0.507	0.512	0.534
	Recall	0.516	0.574	0.822	0.824
k-NN	F1-Score	0.508	0.599	0.894	**0.875**
	Precision	0.665	0.740	1.000	1.000
	Recall	0.504	0.563	0.825	0.800
LR	F1-Score	0.499	0.538	0.894	0.833
	Precision	0.499	0.999	1.000	0.924
	Recall	0.500	0.520	0.825	0.775
XGB	F1-Score	0.499	0.538	0.750	**0.812**
	Precision	0.499	0.999	0.833	0.917
	Recall	0.500	0.520	0.700	0.750

Table 4.20: CERT r4.2 Session Stratified results for GP

Table 4.20 presents the performance of machine learning models on the CERT r4.2 Session dataset using Genetic Programming (GP) with and without Fitness Sharing (FS and NFS). Among the models, k-NN with GP-NFS-10 achieves an F1-Score of 0.599 and Precision of 0.740 for the 0vsAll classification, compared to its performance with the original 127 features (F1-Score of 0.499 and Precision of 0.499). For the 0vs1_3 classification, k-NN achieves an F1-Score of 0.894 and Precision of 1.000, closely matching its performance with the original features (F1-Score of 0.500 and Precision of 0.500).

DT with GP-NFS-10 achieves an F1-Score of 0.655 and Precision of 0.733 for 0vsAll, and an F1-Score of 0.833 and Precision of 0.875 for 0vs1_3. RF with GP-NFS-10 shows an F1-Score of 0.653 and Precision of 0.716 for 0vsAll, and an F1-Score of 0.853 and Precision of 0.929 for 0vs1_3. These results suggest that GP, particularly with NFS, can effectively reduce the feature count to 8, 10, 11 and 13 features, while maintaining a substantial level of performance. This performance is comparable to that achieved with the original and PCA/ICA-reduced feature sets in the case of 0vs1_3 scenarios.

4.6 Summary

This section evaluated the performance of various machine learning models using different feature extraction and dimensionality reduction techniques on the CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2 datasets. The aim was to determine the impact of feature reduction on the models' ability to detect insider threats.

For the CIC-IDS2017 dataset, XGB and RF consistently performed well. Using the original features, XGB achieved an F1-Score of 0.979 and Precision of 0.989. Among reduced feature sets, ICA-20 showed the best performance, with XGB achieving an F1-Score of 0.919 and Precision of 0.963. PCA-20 and GP-FS also retained high performance.

In the CSE-CIC-IDS2018 dataset, RF and XGB maintained high detection performance with PCA-20, achieving F1-Scores and Precision close to their original feature set results. ICA-20 also yielded comparable results.

For the CERT r4.2 dataset, PCA-20 and ICA-20 demonstrated high performance across different granularity levels (Week, Day, and Session). However, reductions beyond 20 components generally led to declines in performance. Genetic Programming (GP) without Fitness Sharing (GP-NFS) often outperformed GP with Fitness Sharing (GP-FS), particularly in broader threat detection scenarios.

Overall, PCA, ICA, and GP effectively reduced feature dimensionalities while maintaining high detection performances in many cases. However, there were variations in performance depending on the technique and dataset. For detailed results of the datasets using different sampling techniques, please refer to Appendix A for results on undersampling and Appendix B for oversampling conditions.

Chapter 5

Conclusions and Future Works

This thesis has systematically explored the role of dimensionality reduction and sampling techniques in enhancing the detection of insider threats within filtration scenarios. The research specifically investigated how different dimensionality reduction techniques, namely Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Genetic Programming with and without Fitness Sharing (GP-FS and GP-NFS), impact the performance of machine learning models. These models were applied to three datasets, CIC-IDS2017, CSE-CIC-IDS2018, and CERT r4.2 (with varying granularities of Week, Day, and Session levels) using three sampling methods, including stratified sampling, undersampling, and oversampling. The research goal is to explore and evaluate how much (if any) these techniques effect the performance of detecting insider threats using infiltration and exfiltration attacks. To achieve this goal, different combinations of sampling and dimensionality reduction techniques are integrated and their performances are compared to the original dimensions on the aforemtioned three network traffic datasets.

Results of this research reveal that models trained on reduced feature sets often yield comparable performances to those trained on original (full) feature sets. Specifically, the research demonstrated that in scenarios of both infiltration and exfiltration attacks, reduced feature sets facilitated by selected techniques allow for efficient data processing while maintaining effective threat detection. This outcome is pivotal for threat hunting teams and security operations centers, which potentially could benefit from deploying lighter, faster, and effective models. The following summarizes the new contributions achieved in this thesis:

- **Exploring Sampling Techniques:** The investigation into the impact of sampling techniques on model performance revealed significant findings across multiple datasets and machine learning models. Stratified sampling consistently provided the best outcomes, enhancing model robustness across all scenarios and datasets. Notably, in the CIC-IDS2017 dataset, stratified sampling enabled models to maintain high detection performance with reduced features. For instance, with original features, RF and XGB

models achieved F1-Scores and Precision above 0.97. When PCA-20 was applied, RF's performance was an F1-Score of 0.90 and Precision of 0.95, while XGB showed an F1-Score of 0.87 and Precision of 0.93. Similarly, ICA-20 and Genetic Programming (GP-FS and GP-NFS) also demonstrated commendable performances, with ICA-20 enabling RF to nearly match its full feature set performance with an F1-Score of 0.94 and Precision of 0.97. XGB maintained a high F1-Score of 0.92 and Precision of 0.96 with ICA-20. Even under the more aggressive feature reduction with GP, XGB achieved an F1-Score of 0.83 and Precision of 0.95 using GP-FS. These patterns were consistent across the CSE-CIC-IDS2018 and CERT r4.2 datasets, where stratified sampling helped retain high detection rates under various dimensionality reduction conditions, confirming the efficacy of this technique in managing dataset imbalances and enhancing the overall accuracy and reliability of insider threat detection systems.

- **Exploring the Dimensionality Reduction Techniques:** This exploration focuses on evaluating the effects of Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Genetic Programming (GP) on the performance of machine learning models across three key datasets. PCA and ICA were implemented with reduced feature sets, namely PCA-20 and ICA-20, significantly maintaining high model performance levels, often achieving near-original performance metrics. For instance, in the CSE-CIC-IDS2018 dataset, both RF and XGB models retained F1-score and Precision of 0.99 under PCA-20 and ICA-20 conditions, demonstrating minimal performance degradation despite the reduced complexity. Similarly, in the CERT r4.2 dataset, although there was a slight decrease in performance under PCA-20 and ICA-20 conditions with XGB's F1-score dropping to 0.65 and 0.73 respectively, these techniques still provided a substantial benefit in reducing feature dimensionality without overly compromising detection capabilities.

  Moreover, Genetic Programming, both with and without Fitness Sharing (GP-FS and GP-NFS), showed a remarkable ability to reduce features drastically to a range of 8 to 15 features while maintaining commendable model efficacy. For example, in the CIC-IDS2017 dataset, XGB managed an F1-score of 0.83 and Precision of 0.95 using GP-FS, underscoring the potential of GP-based methods to provide substantial dimensionality reduction with a lesser but acceptable impact on model performance.

- **Comparison of Reduced Feature Sets:** This analysis focuses on comparing the performance of machine learning models trained with reduced feature sets to those trained with original feature sets, particularly using Principal Component Analysis (PCA) and Independent Component Analysis (ICA) across various dimensionalities (from PCA-5 to PCA-20 and ICA-5 to ICA-20). The comparison underscores the efficacy of dimensionality reduction in maintaining competitive model performances without a full feature set, providing a nuanced view of performance trade-offs at different levels of feature reduction.

  In the CIC-IDS2017 and CSE-CIC-IDS2018 datasets, models like Random Forest (RF) and XGBoost (XGB) showed remarkable adaptability to reduced feature sets. For example, when employing PCA-20, RF and XGB almost mirrored their performance with the original feature set, achieving F1-Scores and Precision rates close to 0.99 in the CSE-CIC-IDS2018 dataset. However, at lower dimensionalities such as PCA-5 and ICA-5, there was a more noticeable decline in performance, indicating that while substantial reductions are possible, they come with diminishing returns in model accuracy and precision.

  Further exploration in the CERT r4.2 dataset, particularly with granularity levels like Week, Day, and Session, revealed more pronounced effects of feature reduction on model performance. For instance, XGB's performance at ICA-20 maintained high precision but saw a reduction in F1-Score from 0.97 to as low as 0.64 in more challenging 0vsAll scenarios, highlighting the critical balance between feature reduction and the ability to detect broader threat scenarios effectively.

- **Granularity-Based Analysis:** This section explores the impact of different data granularities —Week, Day, and Session— on the detection performance of various machine learning models across the CERT r4.2 dataset. The findings of this thesis support those reported by Le et al. [8], i.e. a model's effectiveness depends on the data's temporal resolution. For instance, at the Week level granularity, RF and XGB demonstrated exceptional consistency, maintaining high Precision and F1-Scores close to 0.99 across various configurations. This suggests robustness in a coarser granularity level. However, at the Day level granularity, there was a notable decrease in performance with more aggressive dimensionality reduction techniques like GP-FS, where XGB's

F1-Score dropped to 0.64 in broader threat detection scenarios (0vsAll). Also, at the Session level granularity, again a notable decrease in the performance was observed, specifically with PCA-20 and ICA-20, where XGB's F1-Score reduced to as low as 0.65 in 0vsAll scenarios, significantly impacting the model's ability to generalize across broader threat landscapes. These findings are crucial for security operations centres and threat hunting teams as they suggest that finer granularities might require careful consideration of the balance between dimensionality reduction and model accuracy to ensure effective insider threat detection.

- **Scenario-Based Model Analysis:** The effectiveness of machine learning models in specific detection scenarios, particularly comparing the 0vs1_3 (benign vs. data exfiltration and IT sabotage) with the 0vsAll (benign vs. all scenarios including intellectual property theft), was thoroughly analyzed across the CERT r4.2 dataset. This comparative study highlighted the high performance of models in the 0vs1_3 scenarios, where models were able to achieve more accurate and consistent detection rates compared to the broader 0vsAll scenario.

  Specifically, models like RF and XGB demonstrated substantial efficacy in 0vs1_3 scenarios, consistently maintaining high F1-Scores and Precision. For example, in the Week granularity setting of the CERT r4.2 dataset, RF managed to maintain a high F1-Score of 0.978 and Precision of 1.000 under PCA-20, showcasing its robustness in targeted RF's F1-Score was 0.691 under PCA-20 in broader threat detection scenarios, indicating a potential loss in the ability to generalize across more diverse threat types.

  This pattern was consistently observed across other granularity levels and datasets, reinforcing the notion that while dimensionality reduction techniques such as PCA and ICA can be effective in targeted scenarios, their application in scenarios such as intellectual property theft requires careful calibration and potentially different modeling strategies to maintain effectiveness.

Furthermore, the comprehensive analysis across all datasets and feature reduction settings has shown that PCA-20 features and ICA-20 features consistently delivered high performance across RF, and XGB models. These techniques offer dimensionality reduction without losing much in terms of detection capability, making them suitable for scenarios where operational efficiency is critical. Whereas, GP-NFS offers a unique advantage by reducing the feature

set, which may be more suitable in situations where a smaller and condensed feature set is necessary, rather than mapping the features to a lower dimensionality. This capability can be particularly beneficial when original features need to be minimized to speed up model training phase, while maintaining model effectiveness.

Ultimately, the choice between using PCA, ICA, or GP depends on the specific needs of the security operations centre and threat hunting team. This thesis has laid out a framework for a foundation to explore and evaluate different pipelines, namely integrating sampling, and dimensionality reduction techniques, to better understand which combination yields the best results for detecting and classifying insider threats. The aim is to enable security experts to make more informed decisions on sampling methods and whether to prioritize dimensionality reduction or feature count reduction to select the most appropriate combination for their specific situation in the context of insider threat detection.

In summary, this thesis has set the groundwork for further exploration into advanced feature extraction and dimensionality reduction techniques for filtration attacks in the insider threat detection field. Future research could focus on several promising directions as the following:

- **Sampling Methodologies:** Diversify the sampling methodologies used in model training. Future studies would consider different techniques like Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalances, which are prevalent in cybersecurity datasets. Additionally, varying the data splits, using different ratios or cross-validation methods, could help ensure the models are robust and not overfitting to specific data nuances.

- **Dimensionality Reduction Techniques:** Expand the scope of dimensionality reduction techniques. Systematic testing of methods such as Recursive Feature Elimination (RFE) and Linear Discriminant Analysis (LDA) could ascertain their effectiveness in preserving essential features while reducing redundancy.

- **GP Optimization:** Continue to explore and optimize the application of GP for feature reduction. A deeper exploration into the tuning of GP parameters, such as population size, mutation rates, and crossover probabilities, could uncover more efficient ways to reduce feature dimensions while maintaining or enhancing model accuracy. Additionally, a nuanced approach to fitness sharing and non-fitness sharing within GP

could be examined to determine the best strategy for feature selection in various threat detection scenarios.

- **Operational Efficiency:** Extend the evaluation of machine learning models by incorporating other indicators that measure efficiency alongside performance. Future research could include metrics such as training and testing time to assess the practical performance of models. This dual approach will allow for a comprehensive comparison of reduced versus original feature sets, evaluating not only their performances, but also their operational efficiency, which is crucial for real-time threat detection applications.

- **Data Granularities:** Explore varying levels of data granularity. While this research primarily focused on week, day, and session levels, finer granularities such as minute-level data or broader aggregates such as monthly data could affect the effectiveness of threat detection models. Determining the optimal granularity for model training could improve the model performance.

- **Deep Neural Network Models:** Study the utilization of deep neural network architectures with a focus on unsupervised learning techniques to improve the classification rate for insider threat detection across various scenarios. To this end, Autoencoder models are a good starting point to utilize Latent Space Representation for feature extraction and assess their effectiveness and efficiency in modeling complex data relationships.

- **Generalization of the ML models:** Investigate the generalization and robustness capabilities of machine learning models across diverse contexts to assess adaptability.

- **Access Control Policy Integration:** Explore the inclusion of datasets featuring organizational access control policies to determine their impact on threat detection models.

Bibliography

[1] M. L. Collins et al. Common sense guide to mitigating insider threats, fifth edition. Technical Report CMU/SEI-2015-TR-010, The CERT Insider Threat Center, 2016.

[2] Code42. 2024 annual data exposure report. Technical report, Code42, 2024.

[3] Ponemon Institute. 2020 cost of insider threats: Global report. Technical report, Ponemon Institute, 2020.

[4] Duc C. Le, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Dynamic insider threat detection based on adaptable genetic programming. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2579–2586, 2019.

[5] Minhaj Khan and Mohd. Haroon. Artificial neural network-based intrusion detection in cloud computing using cse-cic-ids2018 datasets. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–4, 2023.

[6] 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, 2020.

[7] Madhu Raut, Sunita Dhavale, Amarjit Singh, and Atul Mehra. Insider threat detection using deep learning: A review. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 856–863, 2020.

[8] Duc C. Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021.

[9] Toya Acharya, Ishan Khatri, Annamalai Annamalai, and Mohamed F Chouikha. Efficacy of machine learning-based classifiers for binary and multi-class network intrusion detection. In *2021 IEEE International Conference on Automatic Control Intelligent Systems (I2CACIS)*, pages 402–407, 2021.

[10] Duc C. Le, Nur Zincir-Heywood, and Malcolm Heywood. Training regime influences to semi-supervised learning for insider threat detection. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 13–18, 2021.

[11] Ishita Karna, Aniket Madam, Chinmay Deokule, Rahul Adhao, and Vinod Pachghare. Ensemble-based filter feature selection technique for building flow-based ids. In *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, pages 324–328, 2021.

[12] Efthimios Pantelidis, Gueltoum Bendiab, Stavros Shiaeles, and Nicholas Kolokotronis. Insider threat detection using deep autoencoder and variational autoencoder neural networks. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 129–134, 2021.

[13] Panpan Zheng, Shuhan Yuan, and Xintao Wu. Using dirichlet marked hawkes processes for insider threat detection. *Digital Threats*, 3(1), oct 2021.

[14] Yangmin Li, Xinhang Yuan, and Wengen Li. An extreme semi-supervised framework based on transformer for network intrusion detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 4204–4208, New York, NY, USA, 2022. Association for Computing Machinery.

[15] Tianci Xu and Peng Zhou. Feature extraction for payload classification: A byte pair encoding algorithm. In *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*, pages 1–5, 2022.

[16] S. D. Erokhin, B. B. Borisenko, I. D. Martishin, and A. S. Fadeev. The dataset features selection for detecting and classifying network attacks. In *2022 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pages 1–8, 2022.

[17] Idio Guarino, Giampaolo Bovenzi, Davide Di Monda, Giuseppe Aceto, Domenico Ciuonzo, and Antonio Pescapé. On the use of machine learning approaches for the early classification in network intrusion detection. In *2022 IEEE International Symposium on Measurements Networking (MN)*, pages 1–6, 2022.

[18] Ruizhe Zhao, Yingxue Mu, Long Zou, and Xiumei Wen. A hybrid intrusion detection system based on feature selection and weighted stacking classifier. *IEEE Access*, 10:71414–71426, 2022.

[19] Sobia Arshad, Rida Zanib, Adeel Akram, Ali Haider, Talha Saeed, and Muhammad Shaheem Raza. Ml-ibotd: Machine learning based intelligent botnet detection. In *2023 3rd International Conference on Artificial Intelligence (ICAI)*, pages 214–219, 2023.

[20] Michal Turčaník. Comparison of evolutionary data clustering of network attacks. In *2023 International Conference on Military Technologies (ICMT)*, pages 1–6, 2023.

[21] Sanchit Singh and Pratik Chattopadhyay. Hierarchical classification using ensemble of feed-forward networks for insider threat detection from activity logs. In *2023 IEEE 20th India Council International Conference (INDICON)*, pages 782–787, 2023.

[22] Simon Bertrand, Josée Desharnais, and Nadia Tawbi. Unsupervised user-based insider threat detection using bayesian gaussian mixture models. In *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, pages 1–10, 2023.

[23] Sahrul Mulia Siregar, Yudha Purwanto, and Suryo Adhi Wibowo. Enhancing network anomaly detection with optimized one-class svm (ocsvm). In *2023 3rd International Conference on Intelligent Cybernetics Technology Applications (ICICyTA)*, pages 84–88, 2023.

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

[25] F. L. Greitzer and R. E. Hohimer. Modeling human behavior to anticipate insider attacks. *Journal of Strategic Security*, 4(2), 2011.

[26] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan. Contemporary sequential network attacks prediction using hidden Markov model. In *Proceedings of the 2019 International Conference on Privacy, Security and Trust (PST)*, August 2019.

[27] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer New York, NY, 2002.

[28] Te-Won Lee. *Independent Component Analysis: Theory and Applications*. Springer New York, NY, 1998.

[29] William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer Berlin Heidelberg, 2002.

[30] Markus Brameier and Wolfgang Banzhaf. *Linear Genetic Programming*. Genetic and Evolutionary Computation. Springer US, Boston, MA, USA, 2007.

[31] Markus Brameier and Wolfgang Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–407, 2001.

[32] John A. Doucette, Andrew R. McIntyre, Peter Lichodzijewski, and Malcolm I. Heywood. Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13(1):71–101, 2012.

[33] Peter Lichodzijewski and Malcolm I. Heywood. Symbiosis, complexification and simplicity under GP. In Martin Pelikan and Jürgen Branke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 853–860. ACM, 2010.

[34] Steven K. Thompson. *Sampling*. John Wiley & Sons, 2012.

[35] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[36] Sufal Das. A new technique for classification method with imbalanced training data. *International Journal of Information Technology*, 16:2177–2185, 2024.

[37] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2019.

[38] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *Forest*, 23, 11 2001.

[39] Inc. O'Reilly Media. *Machine Learning Algorithms*. O'Reilly Media, Inc., 2017.

[40] Chris Albon. *Machine Learning with Python Cookbook*. O'Reilly Media, Inc., 2018.

[41] Joseph M. Hilbe. *Logistic Regression Models*. Chapman and Hall/CRC, 2009.

[42] Anonymous. *Effective XGBoost: Optimizing, Tuning, Understanding, and Deploying*. MetaSnake Publishing, 2021.

Appendix A

Results Using Undersampling Technique

A.1 CIC-IDS2017 Undersampled

A.1.1 CIC-IDS2017 Original with Undersampling

ML Model	F1-Score	Precision	Recall
DT	0.966	0.956	0.977
RF	0.964	0.960	0.969
Gaussian NB	0.186	0.592	0.506
k-NN	0.845	0.822	0.879
LR	0.563	0.599	0.652
XGB	0.976	0.975	0.976

Table A.1: CIC-IDS2017 Undersampled results for the Original

A.1.2 CIC-IDS2017 PCA Undersampled Results

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.779	0.774	0.753	0.792
	Precision	0.755	0.751	0.732	0.768
	Recall	0.833	0.829	0.807	0.842
RF	F1-Score	0.851	0.859	0.848	0.877
	Precision	0.835	0.843	0.836	0.866
	Recall	0.872	0.878	0.862	0.891
Gaussian NB	F1-Score	0.446	0.467	0.502	0.568
	Precision	0.579	0.583	0.590	0.605
	Recall	0.602	0.614	0.630	0.662
k-NN	F1-Score	0.815	0.815	0.824	0.837
	Precision	0.792	0.790	0.799	0.811
	Recall	0.857	0.860	0.868	0.879
LR	F1-Score	0.657	0.758	0.763	0.766
	Precision	0.647	0.769	0.776	0.782
	Recall	0.697	0.748	0.752	0.753
XGB	F1-Score	0.834	0.837	0.832	0.848
	Precision	0.829	0.829	0.821	0.836
	Recall	0.839	0.844	0.845	0.862

Table A.2: CIC-IDS2017 Undersampled results for PCA

A.1.3 CIC-IDS2017 ICA Undersampled Results

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.797	0.836	0.841	0.863
	Precision	0.772	0.810	0.815	0.837
	Recall	0.849	0.882	0.886	0.903
RF	F1-Score	0.863	0.900	0.904	0.927
	Precision	0.846	0.887	0.894	0.923
	Recall	0.886	0.915	0.915	0.930
Gaussian NB	F1-Score	0.309	0.385	0.382	0.460
	Precision	0.562	0.568	0.568	0.581
	Recall	0.544	0.573	0.571	0.609
k-NN	F1-Score	0.816	0.820	0.828	0.837
	Precision	0.792	0.796	0.803	0.812
	Recall	0.858	0.864	0.871	0.879
LR	F1-Score	0.632	0.759	0.769	0.766
	Precision	0.634	0.770	0.789	0.782
	Recall	0.695	0.749	0.754	0.753
XGB	F1-Score	0.835	0.862	0.868	0.899
	Precision	0.823	0.848	0.856	0.893
	Recall	0.850	0.879	0.881	0.906

Table A.3: CIC-IDS2017 Undersampled results for ICA

A.1.4 CIC-IDS2017 GP Undersampled Results

ML Model	Metric	GP-FS	GP-NFS
DT	F1-Score	0.741	0.763
	Precision	0.726	0.758
	Recall	0.765	0.769
RF	F1-Score	0.764	0.762
	Precision	0.755	0.756
	Recall	0.776	0.768
Gaussian NB	F1-Score	0.173	0.225
	Precision	0.532	0.554
	Recall	0.501	0.515
k-NN	F1-Score	0.753	0.641
	Precision	0.743	0.713
	Recall	0.765	0.621
LR	F1-Score	0.170	0.737
	Precision	0.269	0.734
	Recall	0.500	0.741
XGB	F1-Score	0.828	0.824
	Precision	0.919	0.920
	Recall	0.782	0.778

Table A.4: CIC-IDS2017 Undersampled results for GP-FS and GP-NFS

A.2 CSE-CIC-IDS2018 Undersampled

A.2.1 CSE-CIC-IDS2018 Original with Undersampling

ML Model	F1-Score	Precision	Recall
DT	0.997	0.996	0.998
RF	0.999	0.999	0.999
Gaussian NB	0.183	0.602	0.507
k-NN	0.962	0.957	0.969
LR	0.634	0.682	0.782
XGB	0.999	0.999	0.999

Table A.5: CSE-CIC-IDS2018 Original with Undersampling

A.2.2 CSE-CIC-IDS2018 PCA Undersampled Results

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.971	0.948	0.978	0.951
	Precision	0.962	0.956	0.971	0.961
	Recall	0.981	0.939	0.984	0.942
RF	F1-Score	0.989	0.993	0.993	0.995
	Precision	0.990	0.995	0.994	0.996
	Recall	0.989	0.992	0.992	0.994
Gaussian NB	F1-Score	0.944	0.947	0.948	0.949
	Precision	0.951	0.956	0.958	0.959
	Recall	0.937	0.939	0.939	0.940
k-NN	F1-Score	0.984	0.989	0.991	0.991
	Precision	0.981	0.988	0.990	0.989
	Recall	0.987	0.991	0.992	0.993
LR	F1-Score	0.941	0.942	0.945	0.944
	Precision	0.940	0.941	0.945	0.941
	Recall	0.941	0.944	0.946	0.946
XGB	F1-Score	0.985	0.991	0.991	0.993
	Precision	0.984	0.991	0.991	0.993
	Recall	0.986	0.991	0.992	0.993

Table A.6: CSE-CIC-IDS2018 Undersampled results for PCA

A.2.3 CSE-CIC-IDS2018 ICA Undersampled Results

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.975	0.983	0.985	0.985
	Precision	0.967	0.978	0.979	0.980
	Recall	0.984	0.989	0.990	0.990
RF	F1-Score	0.991	0.995	0.995	0.995
	Precision	0.992	0.996	0.996	0.996
	Recall	0.990	0.994	0.994	0.995
Gaussian NB	F1-Score	0.932	0.904	0.841	0.877
	Precision	0.929	0.887	0.813	0.855
	Recall	0.935	0.923	0.895	0.908
k-NN	F1-Score	0.985	0.990	0.991	0.991
	Precision	0.982	0.988	0.990	0.989
	Recall	0.987	0.991	0.992	0.993
LR	F1-Score	0.936	0.942	0.952	0.953
	Precision	0.932	0.941	0.962	0.963
	Recall	0.939	0.944	0.943	0.944
XGB	F1-Score	0.987	0.993	0.994	0.994
	Precision	0.987	0.993	0.994	0.994
	Recall	0.988	0.993	0.994	0.994

Table A.7: CSE-CIC-IDS2018 Undersampled results for ICA

A.2.4 CSE-CIC-IDS2018 GP Undersampled Results

ML Model	Metric	GP-FS	GP-NFS
DT	F1-Score	0.975	0.978
	Precision	0.968	0.972
	Recall	0.982	0.985
RF	F1-Score	0.983	0.984
	Precision	0.980	0.981
	Recall	0.986	0.987
Gaussian NB	F1-Score	0.287	0.444
	Precision	0.608	0.432
	Recall	0.558	0.500
k-NN	F1-Score	0.983	0.984
	Precision	0.981	0.983
	Recall	0.985	0.985
LR	F1-Score	0.471	0.690
	Precision	0.628	0.705
	Recall	0.662	0.818
XGB	F1-Score	0.983	0.986
	Precision	0.985	0.988
	Recall	0.982	0.984

Table A.8: CSE-CIC-IDS2018 Undersampled results for GP-FS and GP-NFS

A.3 CERT r4.2 Week Undersampled

A.3.1 CERT r4.2 Week Original with Undersampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.535	0.529	0.954	0.521	0.514	0.985
RF	0.522	0.524	0.946	0.730	0.650	0.999
Gaussian NB	0.495	0.516	0.902	0.509	0.509	0.970
k-NN	0.366	0.503	0.653	0.408	0.501	0.684
LR	0.478	0.512	0.876	0.375	0.500	0.596
XGB	0.533	0.528	0.953	0.520	0.514	0.969

Table A.9: CERT r4.2 Week results for Original with Undersampling

A.3.2 CERT r4.2 Week PCA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.489	0.498	0.503	0.497	0.518	0.588	0.588	0.588
	Precision	0.514	0.516	0.518	0.515	0.514	0.550	0.550	0.550
	Recall	0.859	0.877	0.883	0.875	0.968	0.993	0.993	0.993
RF	F1-Score	0.488	0.488	0.492	0.490	0.506	0.648	0.586	0.598
	Precision	0.515	0.515	0.516	0.515	0.509	0.589	0.548	0.556
	Recall	0.850	0.850	0.858	0.854	0.952	0.996	0.992	0.993
Gaussian NB	F1-Score	0.484	0.503	0.500	0.498	0.482	0.507	0.505	0.506
	Precision	0.504	0.508	0.507	0.507	0.503	0.511	0.510	0.511
	Recall	0.577	0.591	0.602	0.607	0.788	0.977	0.976	0.977
k-NN	F1-Score	0.483	0.489	0.485	0.484	0.507	1.000	1.000	1.000
	Precision	0.514	0.515	0.514	0.513	0.510	1.000	1.000	1.000
	Recall	0.840	0.855	0.844	0.841	0.954	1.000	1.000	1.000
LR	F1-Score	0.488	0.467	0.485	0.486	0.546	1.000	1.000	1.000
	Precision	0.514	0.511	0.514	0.514	0.526	1.000	1.000	1.000
	Recall	0.856	0.806	0.848	0.847	0.984	1.000	1.000	1.000
XGB	F1-Score	0.489	0.489	0.494	0.494	0.515	0.770	0.770	0.770
	Precision	0.514	0.515	0.516	0.516	0.512	0.700	0.700	0.700
	Recall	0.854	0.851	0.863	0.862	0.965	0.999	0.999	0.999

Table A.10: CERT r4.2 Week Stratified results for PCA Undersampled

A.3.3 CERT r4.2 Week ICA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.520	0.494	0.492	0.489	0.547	0.541	0.518	0.487
	Precision	0.523	0.515	0.514	0.513	0.527	0.524	0.514	0.505
	Recall	0.908	0.867	0.868	0.860	0.985	0.982	0.968	0.915
RF	F1-Score	0.511	0.491	0.489	0.488	0.524	0.536	0.511	0.531
	Precision	0.520	0.515	0.515	0.515	0.516	0.522	0.511	0.519
	Recall	0.893	0.856	0.852	0.851	0.973	0.981	0.960	0.978
Gaussian NB	F1-Score	0.502	0.513	0.500	0.512	0.506	0.524	0.490	0.508
	Precision	0.507	0.512	0.508	0.502	0.509	0.512	0.506	0.510
	Recall	0.944	0.957	0.932	0.956	0.957	0.973	0.921	0.956
k-NN	F1-Score	0.486	0.491	0.481	0.479	0.519	1.000	1.000	1.000
	Precision	0.514	0.515	0.513	0.513	0.514	1.000	1.000	1.000
	Recall	0.847	0.856	0.836	0.831	0.969	1.000	1.000	1.000
LR	F1-Score	0.522	0.468	0.484	0.484	0.529	1.000	1.000	1.000
	Precision	0.519	0.511	0.514	0.514	0.518	1.000	1.000	1.000
	Recall	0.718	0.806	0.848	0.848	0.988	1.000	1.000	1.000
XGB	F1-Score	0.514	0.499	0.499	0.500	0.540	0.638	0.509	0.514
	Precision	0.522	0.517	0.517	0.517	0.523	0.582	0.510	0.512
	Recall	0.898	0.873	0.874	0.879	0.981	0.996	0.958	0.964

Table A.11: CERT r4.2 Week Stratified results for ICA Undersampled

A.3.4 CERT r4.2 Week GP Undersampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.519	0.516	0.486	0.443
	Precision	0.523	0.522	0.505	0.502
	Recall	0.905	0.903	0.912	0.783
RF	F1-Score	0.528	0.518	0.978	0.411
	Precision	0.527	0.523	1.000	0.501
	Recall	0.958	0.951	0.958	0.688
Gaussian NB	F1-Score	0.567	0.554	0.488	0.373
	Precision	0.542	0.536	0.504	0.500
	Recall	0.711	0.873	0.876	0.592
k-NN	F1-Score	0.521	0.515	1.000	0.436
	Precision	0.524	0.522	1.000	0.501
	Recall	0.907	0.898	1.000	0.765
LR	F1-Score	0.539	0.538	1.000	0.978
	Precision	0.530	0.530	1.000	1.000
	Recall	0.941	0.934	1.000	0.958
XGB	F1-Score	0.530	0.514	0.805	0.423
	Precision	0.527	0.522	0.729	0.502
	Recall	0.951	0.948	0.958	0.724

Table A.12: CERT r4.2 Week Stratified results for GP Undersampled

A.4 CERT r4.2 Day Undersampled

A.4.1 CERT r4.2 Day Original with Undersampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.523	0.520	0.935	0.457	0.501	0.894
RF	0.539	0.527	0.951	0.509	0.507	0.980
Gaussian NB	0.480	0.508	0.866	0.491	0.502	0.952
k-NN	0.377	0.501	0.593	0.403	0.500	0.673
LR	0.449	0.505	0.781	0.476	0.501	0.898
XGB	0.563	0.538	0.966	0.488	0.502	0.936

Table A.13: CERT r4.2 Day Original with Undersampling

A.4.2 CERT r4.2 Day PCA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.475	0.474	0.469	0.471	0.454	0.447	0.458	0.442
	Precision	0.508	0.508	0.507	0.507	0.501	0.501	0.501	0.501
	Recall	0.849	0.846	0.834	0.838	0.826	0.803	0.839	0.789
RF	F1-Score	0.486	0.493	0.492	0.490	0.490	0.492	0.494	0.497
	Precision	0.510	0.511	0.511	0.511	0.503	0.503	0.503	0.504
	Recall	0.873	0.888	0.886	0.882	0.943	0.948	0.953	0.960
Gaussian NB	F1-Score	0.499	0.501	0.502	0.497	0.494	0.499	0.501	0.495
	Precision	0.503	0.504	0.504	0.503	0.503	0.504	0.505	0.503
	Recall	0.964	0.967	0.970	0.957	0.953	0.954	0.968	0.956
k-NN	F1-Score	0.484	0.473	0.466	0.466	0.486	0.493	0.495	0.497
	Precision	0.509	0.508	0.507	0.507	0.502	0.503	0.503	0.504
	Recall	0.871	0.840	0.824	0.822	0.931	0.951	0.956	0.959
LR	F1-Score	0.464	0.454	0.461	0.467	0.506	0.528	0.529	0.498
	Precision	0.505	0.505	0.506	0.507	0.506	0.516	0.516	0.504
	Recall	0.829	0.797	0.814	0.829	0.976	0.991	0.991	0.961
XGB	F1-Score	0.486	0.491	0.492	0.493	0.484	0.485	0.489	0.480
	Precision	0.510	0.511	0.511	0.512	0.502	0.502	0.503	0.502
	Recall	0.873	0.884	0.887	0.888	0.924	0.928	0.939	0.913

Table A.14: CERT r4.2 Day Stratified results for PCA

A.4.3 CERT r4.2 Day ICA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.470	0.469	0.479	0.473	0.449	0.456	0.440	0.461
	Precision	0.507	0.507	0.509	0.507	0.501	0.501	0.501	0.501
	Recall	0.839	0.835	0.859	0.843	0.811	0.832	0.780	0.850
RF	F1-Score	0.489	0.495	0.497	0.490	0.495	0.498	0.494	0.502
	Precision	0.511	0.512	0.513	0.511	0.503	0.504	0.503	0.505
	Recall	0.880	0.893	0.895	0.882	0.955	0.962	0.953	0.970
Gaussian NB	F1-Score	0.500	0.503	0.500	0.501	0.494	0.496	0.493	0.488
	Precision	0.503	0.504	0.503	0.504	0.503	0.504	0.503	0.502
	Recall	0.973	0.973	0.968	0.964	0.953	0.952	0.949	0.934
k-NN	F1-Score	0.482	0.466	0.461	0.461	0.486	0.491	0.496	0.498
	Precision	0.509	0.507	0.507	0.507	0.502	0.503	0.503	0.504
	Recall	0.868	0.824	0.810	0.838	0.931	0.951	0.958	0.962
LR	F1-Score	0.501	0.454	0.458	0.466	0.505	0.526	0.529	0.511
	Precision	0.504	0.505	0.506	0.507	0.506	0.515	0.519	0.508
	Recall	0.968	0.796	0.806	0.829	0.953	0.991	0.953	0.982
XGB	F1-Score	0.487	0.495	0.500	0.496	0.477	0.484	0.486	0.488
	Precision	0.510	0.512	0.513	0.512	0.502	0.502	0.503	0.502
	Recall	0.876	0.891	0.903	0.894	0.901	0.925	0.932	0.936

Table A.15: CERT r4.2 Day Stratified results for ICA

A.4.4 CERT r4.2 Day GP Undersampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.495	0.524	0.447	0.502
	Precision	0.511	0.521	0.501	0.505
	Recall	0.896	0.936	0.802	0.970
RF	F1-Score	0.495	0.523	0.444	0.505
	Precision	0.512	0.521	0.501	0.506
	Recall	0.893	0.936	0.796	0.974
Gaussian NB	F1-Score	0.494	0.503	0.396	0.519
	Precision	0.509	0.508	0.500	0.511
	Recall	0.906	0.948	0.653	0.987
k-NN	F1-Score	0.497	0.522	0.463	0.768
	Precision	0.511	0.520	0.501	0.690
	Recall	0.906	0.936	0.856	0.999
LR	F1-Score	0.493	0.501	0.509	0.550
	Precision	0.509	0.512	0.507	0.527
	Recall	0.901	0.912	0.980	0.995
XGB	F1-Score	0.495	0.531	0.459	0.527
	Precision	0.512	0.524	0.501	0.515
	Recall	0.895	0.946	0.856	0.991

Table A.16: CERT r4.2 Day Stratified results for GP

A.5 CERT r4.2 Session Undersampled

CERT r4.2 Session Original with Undersampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.4997	0.5119	0.9212	0.4691	0.5008	0.9138
RF	0.5161	0.5171	0.9582	0.4982	0.5032	0.9591
Gaussian NB	0.3955	0.5024	0.7392	0.4672	0.5006	0.8362
k-NN	0.3673	0.5015	0.6575	0.4048	0.5002	0.7143
LR	0.4507	0.5038	0.7646	0.4601	0.5005	0.8243
XGB	0.5448	0.5289	0.9785	0.4902	0.5019	0.9485

Table A.17: CERT r4.2 Session Original with Undersampling

CERT r4.2 Session PCA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.4742	0.4789	0.4741	0.4723	0.4848	0.4897	0.4897	0.4931
	Precision	0.5068	0.5073	0.5069	0.5067	0.5013	0.5018	0.5018	0.5023
	Recall	0.8597	0.8594	0.8703	0.8701	0.8906	0.9230	0.9478	0.9527
RF	F1-Score	0.4924	0.4987	0.4918	0.4916	0.4944	0.4903	0.4930	0.4997
	Precision	0.5100	0.5116	0.5101	0.5099	0.5024	0.5019	0.5022	0.5035
	Recall	0.9010	0.9202	0.9110	0.9043	0.9544	0.9487	0.9526	0.9607
Gaussian NB	F1-Score	0.4932	0.4970	0.4934	0.4924	0.5215	0.5038	0.5109	0.5129
	Precision	0.5004	0.5016	0.5034	0.5039	0.5118	0.5041	0.5068	0.5077
	Recall	0.5066	0.5190	0.5771	0.6003	0.8222	0.9831	0.9900	0.9911
k-NN	F1-Score	0.4824	0.4720	0.4708	0.4650	0.4801	0.4827	0.4866	0.4788
	Precision	0.5082	0.5068	0.5068	0.5063	0.5012	0.5013	0.5016	0.5011
	Recall	0.8916	0.8783	0.8875	0.8841	0.9324	0.9366	0.9430	0.9302
LR	F1-Score	0.4512	0.4491	0.4582	0.4611	0.4814	0.4863	0.4881	0.4916
	Precision	0.5035	0.5042	0.5049	0.5052	0.5012	0.5015	0.5017	0.5021
	Recall	0.7437	0.8014	0.8163	0.8291	0.9345	0.9350	0.9453	0.9512
XGB	F1-Score	0.4830	0.4897	0.4891	0.4929	0.4929	0.4883	0.4894	0.4966
	Precision	0.5084	0.5095	0.5095	0.5103	0.5020	0.5017	0.5018	0.5112
	Recall	0.8965	0.9002	0.9037	0.9121	0.9561	0.9413	0.9445	0.9542

Table A.18: CERT r4.2 Session Undersampled results for PCA

CERT r4.2 Session ICA Undersampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.4739	0.4804	0.4791	0.4774	0.4809	0.4916	0.4678	0.4905
	Precision	0.5070	0.5078	0.5073	0.5072	0.5012	0.5021	0.5008	0.5019
	Recall	0.8744	0.8827	0.8575	0.8681	0.9337	0.9507	0.9117	0.9489
RF	F1-Score	0.4969	0.4928	0.4933	0.4913	0.4921	0.4945	0.4932	0.4903
	Precision	0.5111	0.5103	0.5105	0.5099	0.5021	0.5025	0.5023	0.5019
	Recall	0.9099	0.9142	0.9191	0.9083	0.9514	0.9546	0.9528	0.9487
Gaussian NB	F1-Score	0.4967	0.5017	0.4935	0.4937	0.5049	0.5124	0.5082	0.5060
	Precision	0.5019	0.5025	0.5019	0.5025	0.5044	0.5075	0.5057	0.5048
	Recall	0.5260	0.5166	0.5355	0.5495	0.8173	0.8205	0.8190	0.8180
k-NN	F1-Score	0.4803	0.4665	0.4682	0.4600	0.4757	0.4817	0.4892	0.4826
	Precision	0.5078	0.5061	0.5066	0.5059	0.5010	0.5013	0.5018	0.5013
	Recall	0.8783	0.8667	0.8905	0.8837	0.9249	0.9350	0.9470	0.9365
LR	F1-Score	0.5382	0.4469	0.4583	0.4598	0.4795	0.4835	0.4885	0.4924
	Precision	0.9989	0.5039	0.5049	0.5051	0.5012	0.5014	0.5017	0.5022
	Recall	0.5202	0.7852	0.8166	0.8251	0.9314	0.9345	0.9453	0.9513
XGB	F1-Score	0.4879	0.4919	0.4965	0.4966	0.4817	0.4871	0.4903	0.4927
	Precision	0.5093	0.5103	0.5112	0.5113	0.5013	0.5016	0.5019	0.5022
	Recall	0.8853	0.9218	0.9203	0.9225	0.9350	0.9437	0.9486	0.9521

Table A.19: CERT r4.2 Session Undersampled results for ICA

CERT r4.2 Session GP Undersampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.4854	0.4912	0.5316	0.5316
	Precision	0.5078	0.5099	0.5171	0.5171
	Recall	0.8354	0.9083	0.9721	0.9721
RF	F1-Score	0.4835	0.4906	0.5316	0.5316
	Precision	0.5081	0.5098	0.5171	0.5171
	Recall	0.8715	0.9076	0.9721	0.9721
Gaussian NB	F1-Score	0.4792	0.5058	0.4771	0.4718
	Precision	0.5054	0.5094	0.5011	0.5009
	Recall	0.7445	0.6964	0.9273	0.9185
k-NN	F1-Score	0.4881	0.5099	0.7261	0.5764
	Precision	0.5083	0.5139	0.6484	0.5418
	Recall	0.8450	0.8747	0.9748	0.9739
LR	F1-Score	0.4901	0.4528	0.4780	0.4803
	Precision	0.5082	0.5050	0.5011	0.5012
	Recall	0.8153	0.8500	0.9288	0.9327
XGB	F1-Score	0.4887	0.4975	0.5316	0.5316
	Precision	0.5090	0.5113	0.5171	0.5171
	Recall	0.8798	0.9149	0.9721	0.9721

Table A.20: CERT r4.2 Session Undersampled results for GP

Appendix B

Results Using Oversampling Technique

B.1 CIC-IDS2017 Oversampled

B.1.1 CIC-IDS2017 Original with Oversampling

ML Model	F1-Score	Precision	Recall
DT	0.981	0.981	0.980
RF	0.975	0.986	0.965
Gaussian NB	0.186	0.592	0.506
k-NN	0.862	0.839	0.893
LR	0.563	0.599	0.652
XGB	0.976	0.977	0.976

Table B.1: CIC-IDS2017 Original with Oversampling

B.1.2 CIC-IDS2017 PCA Oversampled Results

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.846	0.841	0.809	0.847
	Precision	0.846	0.839	0.802	0.845
	Recall	0.846	0.843	0.818	0.849
RF	F1-Score	0.890	0.894	0.874	0.903
	Precision	0.919	0.928	0.913	0.938
	Recall	0.867	0.868	0.846	0.876
Gaussian NB	F1-Score	0.389	0.417	0.440	0.476
	Precision	0.571	0.574	0.578	0.585
	Recall	0.576	0.589	0.600	0.618
k-NN	F1-Score	0.823	0.826	0.838	0.853
	Precision	0.799	0.802	0.814	0.829
	Recall	0.862	0.868	0.877	0.889
LR	F1-Score	0.660	0.758	0.763	0.766
	Precision	0.650	0.769	0.778	0.783
	Recall	0.700	0.749	0.752	0.753
XGB	F1-Score	0.836	0.842	0.835	0.854
	Precision	0.832	0.837	0.827	0.847
	Recall	0.841	0.847	0.844	0.862

Table B.2: CIC-IDS2017 Oversampled results for PCA

B.1.3 CIC-IDS2017 ICA Oversampled Results

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.846	0.894	0.898	0.916
	Precision	0.862	0.894	0.898	0.917
	Recall	0.833	0.893	0.897	0.916
RF	F1-Score	0.899	0.930	0.931	0.946
	Precision	0.924	0.951	0.954	0.968
	Recall	0.880	0.911	0.911	0.927
Gaussian NB	F1-Score	0.237	0.344	0.321	0.376
	Precision	0.550	0.563	0.561	0.573
	Recall	0.518	0.556	0.547	0.573
k-NN	F1-Score	0.825	0.838	0.845	0.855
	Precision	0.801	0.814	0.821	0.832
	Recall	0.864	0.876	0.882	0.891
LR	F1-Score	0.633	0.759	0.768	0.766
	Precision	0.634	0.770	0.787	0.783
	Recall	0.695	0.749	0.753	0.753
XGB	F1-Score	0.839	0.864	0.873	0.906
	Precision	0.828	0.851	0.864	0.906
	Recall	0.853	0.879	0.883	0.907

Table B.3: CIC-IDS2017 Oversampled results for ICA

B.1.4 CIC-IDS2017 GP Oversampled Results

ML Model	Metric	GP-FS	GP-NFS
DT	F1-Score	0.795	0.799
	Precision	0.806	0.823
	Recall	0.786	0.781
RF	F1-Score	0.803	0.799
	Precision	0.821	0.821
	Recall	0.789	0.781
Gaussian NB	F1-Score	0.170	0.173
	Precision	0.491	0.535
	Recall	0.500	0.501
k-NN	F1-Score	0.738	0.786
	Precision	0.723	0.805
	Recall	0.762	0.771
LR	F1-Score	0.170	0.737
	Precision	0.269	0.733
	Recall	0.500	0.740
XGB	F1-Score	0.826	0.825
	Precision	0.913	0.923
	Recall	0.782	0.778

Table B.4: CIC-IDS2017 Oversampled results for GP-FS and GP-NFS

B.2 CSE-CIC-IDS2018 Oversampled

B.2.1 CSE-CIC-IDS2018 Original with Oversampling

ML Model	F1-Score	Precision	Recall
DT	0.999	0.999	0.998
RF	0.999	1.000	0.999
Gaussian NB	0.183	0.602	0.507
k-NN	0.959	0.950	0.969
LR	0.634	0.682	0.782
XGB	0.999	1.000	0.999

Table B.5: CSE-CIC-IDS2018 Original with Oversampling

B.2.2 CSE-CIC-IDS2018 PCA Oversampled Results

ML Model	Metric	PCA-5	PCA-10	PCA-15	PCA-20
DT	F1-Score	0.985	0.987	0.984	0.956
	Precision	0.985	0.987	0.983	0.976
	Recall	0.984	0.987	0.985	0.939
RF	F1-Score	0.991	0.992	0.994	0.995
	Precision	0.995	0.996	0.997	0.997
	Recall	0.988	0.988	0.991	0.993
Gaussian NB	F1-Score	0.911	0.942	0.934	0.923
	Precision	0.896	0.946	0.933	0.916
	Recall	0.928	0.938	0.935	0.930
k-NN	F1-Score	0.983	0.989	0.992	0.992
	Precision	0.979	0.987	0.990	0.990
	Recall	0.987	0.992	0.994	0.994
LR	F1-Score	0.940	0.942	0.945	0.944
	Precision	0.939	0.941	0.945	0.942
	Recall	0.942	0.944	0.946	0.946
XGB	F1-Score	0.986	0.980	0.992	0.994
	Precision	0.985	0.988	0.992	0.994
	Recall	0.986	0.973	0.992	0.993

Table B.6: CSE-CIC-IDS2018 Oversampled results for PCA

B.2.3 CSE-CIC-IDS2018 ICA Oversampled Results

ML Model	Metric	ICA-5	ICA-10	ICA-15	ICA-20
DT	F1-Score	0.978	0.992	0.992	0.993
	Precision	0.985	0.993	0.992	0.993
	Recall	0.971	0.992	0.992	0.993
RF	F1-Score	0.993	0.996	0.996	0.996
	Precision	0.995	0.998	0.998	0.998
	Recall	0.990	0.994	0.993	0.994
Gaussian NB	F1-Score	0.835	0.690	0.733	0.838
	Precision	0.806	0.698	0.725	0.810
	Recall	0.895	0.804	0.835	0.888
k-NN	F1-Score	0.984	0.991	0.992	0.993
	Precision	0.980	0.989	0.991	0.991
	Recall	0.988	0.991	0.994	0.994
LR	F1-Score	0.936	0.942	0.952	0.953
	Precision	0.933	0.941	0.962	0.963
	Recall	0.939	0.944	0.943	0.944
XGB	F1-Score	0.988	0.994	0.994	0.995
	Precision	0.988	0.995	0.995	0.995
	Recall	0.988	0.994	0.994	0.995

Table B.7: CSE-CIC-IDS2018 Oversampled results for ICA

B.2.4 CSE-CIC-IDS2018 GP Oversampled Results

ML Model	Metric	GP-FS	GP-NFS
DT	F1-Score	0.985	0.988
	Precision	0.986	0.988
	Recall	0.985	0.988
RF	F1-Score	0.987	0.989
	Precision	0.989	0.990
	Recall	0.986	0.988
Gaussian NB	F1-Score	0.284	0.452
	Precision	0.607	0.634
	Recall	0.556	0.657
k-NN	F1-Score	0.980	0.982
	Precision	0.976	0.980
	Recall	0.984	0.985
LR	F1-Score	0.472	0.688
	Precision	0.628	0.704
	Recall	0.662	0.816
XGB	F1-Score	0.983	0.986
	Precision	0.985	0.988
	Recall	0.982	0.984

Table B.8: CSE-CIC-IDS2018 Oversampled results for GP-FS and GP-NFS

B.3 CERT r4.2 Week Oversampled

B.3.1 CERT r4.2 Week Original with Oversampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.676	0.698	0.658	0.868	1.000	0.792
RF	0.769	0.864	0.714	0.955	1.000	0.917
Gaussian NB	0.501	0.517	0.908	0.493	0.505	0.883
k-NN	0.518	0.514	0.527	0.500	0.500	0.500
LR	0.491	0.515	0.890	0.505	0.508	0.854
XGB	0.895	0.862	0.936	1.000	1.000	1.000

Table B.9: CERT r4.2 Week Original with Oversampling

B.3.2 CERT r4.2 Week PCA Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.592	0.638	0.587	0.612	0.778	0.955	0.955	0.955
	Precision	0.600	0.635	0.588	0.626	0.916	1.000	1.000	1.000
	Recall	0.586	0.641	0.585	0.602	0.708	0.917	0.917	0.917
RF	F1-Score	0.592	0.719	0.681	0.702	0.900	0.955	0.955	0.955
	Precision	0.790	0.972	0.998	0.998	1.000	1.000	1.000	1.000
	Recall	0.555	0.643	0.611	0.627	0.833	0.917	0.917	0.917
Gaussian NB	F1-Score	0.490	0.512	0.512	0.510	0.489	0.536	0.527	0.520
	Precision	0.504	0.511	0.511	0.510	0.504	0.521	0.517	0.514
	Recall	0.570	0.578	0.584	0.596	0.757	0.990	0.988	0.985
k-NN	F1-Score	0.612	0.659	0.662	0.666	0.958	1.000	1.000	1.000
	Precision	0.581	0.615	0.618	0.621	0.958	1.000	1.000	1.000
	Recall	0.686	0.766	0.758	0.766	0.958	1.000	1.000	1.000
LR	F1-Score	0.489	0.478	0.494	0.493	0.547	0.978	0.978	0.978
	Precision	0.514	0.512	0.516	0.516	0.527	1.000	1.000	1.000
	Recall	0.858	0.860	0.908	0.907	0.992	0.958	0.958	0.958
XGB	F1-Score	0.598	0.733	0.736	0.729	0.900	0.955	0.955	0.955
	Precision	0.581	0.736	0.763	0.809	1.000	1.000	1.000	1.000
	Recall	0.624	0.729	0.713	0.682	0.833	0.917	0.917	0.917

Table B.10: CERT r4.2 Week Oversampled results for PCA

B.3.3 CERT r4.2 Week ICA Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	**I-10**	**I-15**	**I-20**	**I-5**	**I-10**	**I-15**	**I-20**
DT	F1-Score	0.644	0.619	0.664	0.621	0.848	0.955	0.909	0.909
	Precision	0.640	0.621	0.671	0.618	0.863	1.000	0.950	0.950
	Recall	0.649	0.617	0.657	0.625	0.833	0.917	0.875	0.875
RF	F1-Score	0.666	0.668	0.670	0.668	0.929	0.955	0.929	0.955
	Precision	0.776	0.962	0.998	0.962	1.000	1.000	1.000	1.000
	Recall	0.619	0.603	0.603	0.603	0.875	0.917	0.875	0.917
Gaussian NB	F1-Score	0.505	0.531	0.519	0.528	0.497	0.552	0.531	0.542
	Precision	0.508	0.521	0.514	0.519	0.505	0.530	0.519	0.524
	Recall	0.586	0.593	0.588	0.606	0.807	0.993	0.989	0.991
k-NN	F1-Score	0.610	0.663	0.662	0.654	0.958	1.000	1.000	1.000
	Precision	0.581	0.617	0.618	0.612	0.958	1.000	1.000	1.000
	Recall	0.678	0.774	0.766	0.750	0.958	1.000	1.000	1.000
LR	F1-Score	0.490	0.477	0.493	0.493	0.546	0.978	1.000	1.000
	Precision	0.512	0.512	0.516	0.516	0.527	1.000	1.000	1.000
	Recall	0.786	0.860	0.907	0.907	0.992	0.958	1.000	1.000
XGB	F1-Score	0.664	0.730	0.690	0.756	0.935	0.955	0.955	0.955
	Precision	0.628	0.740	0.722	0.803	0.955	1.000	1.000	1.000
	Recall	0.727	0.721	0.666	0.722	0.917	0.917	0.917	0.917

Table B.11: CERT r4.2 Week Oversampled results for ICA

B.3.4 CERT r4.2 Week GP Oversampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.565	0.614	0.955	0.955
	Precision	0.541	0.612	1.000	1.000
	Recall	0.741	0.617	0.917	0.917
RF	F1-Score	0.564	0.614	1.000	0.900
	Precision	0.540	0.605	1.000	1.000
	Recall	0.741	0.625	1.000	0.833
Gaussian NB	F1-Score	0.568	0.556	0.488	0.382
	Precision	0.544	0.536	0.505	0.500
	Recall	0.681	0.851	0.876	0.599
k-NN	F1-Score	0.617	0.636	1.000	0.978
	Precision	0.601	0.608	1.000	1.000
	Recall	0.640	0.687	1.000	0.958
LR	F1-Score	0.539	0.540	1.000	0.978
	Precision	0.530	0.530	1.000	1.000
	Recall	0.927	0.927	1.000	0.958
XGB	F1-Score	0.562	0.643	0.978	0.978
	Precision	0.539	0.596	1.000	1.000
	Recall	0.770	0.796	0.958	0.958

Table B.12: CERT r4.2 Week Oversampled results for GP

B.4 CERT r4.2 Day Oversampled

B.4.1 CERT r4.2 Day Original with Oversampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.824	0.860	0.795	0.786	0.857	0.738
RF	0.845	0.990	0.767	0.962	1.000	0.929
Gaussian NB	0.485	0.508	0.800	0.492	0.502	0.763
k-NN	0.508	0.507	0.511	0.500	0.500	0.500
LR	0.451	0.505	0.798	0.471	0.501	0.823
XGB	0.978	0.977	0.979	0.975	1.000	0.952

Table B.13: CERT r4.2 Day Original with Oversampling

B.4.2 CERT r4.2 Day PCA Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	**P-10**	**P-15**	**P-20**	**P-5**	**P-10**	**P-15**	**P-20**
DT	F1-Score	0.586	0.599	0.595	0.584	0.891	0.929	0.961	0.939
	Precision	0.589	0.607	0.596	0.587	0.860	0.929	0.999	0.950
	Recall	0.584	0.592	0.595	0.582	0.929	0.929	0.929	0.929
RF	F1-Score	0.592	0.598	0.602	0.614	0.947	0.962	0.962	0.962
	Precision	0.755	0.786	0.870	0.893	1.000	1.000	1.000	1.000
	Recall	0.557	0.559	0.560	0.567	0.905	0.929	0.929	0.929
Gaussian NB	F1-Score	0.496	0.503	0.501	0.499	0.502	0.512	0.507	0.503
	Precision	0.502	0.504	0.503	0.504	0.505	0.508	0.506	0.505
	Recall	0.534	0.535	0.531	0.542	0.961	0.967	0.977	0.961
k-NN	F1-Score	0.564	0.594	0.605	0.605	0.858	0.888	0.913	0.904
	Precision	0.546	0.570	0.578	0.575	0.797	0.839	0.880	0.865
	Recall	0.608	0.648	0.661	0.676	0.952	0.952	0.952	0.952
LR	F1-Score	0.466	0.460	0.462	0.465	0.519	0.588	0.532	0.517
	Precision	0.505	0.506	0.506	0.506	0.511	0.549	0.518	0.510
	Recall	0.736	0.801	0.810	0.819	0.970	0.951	0.948	0.946
XGB	F1-Score	0.538	0.596	0.615	0.632	0.895	0.962	0.962	0.950
	Precision	0.525	0.560	0.574	0.591	0.886	1.000	1.000	0.974
	Recall	0.750	0.777	0.773	0.746	0.905	0.929	0.929	0.929

Table B.14: CERT r4.2 Day PCA Oversampled results

B.4.3 CERT r4.2 Day ICA Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	I-10	I-15	I-20	I-5	I-10	I-15	I-20
DT	F1-Score	0.599	0.598	0.623	0.588	0.915	0.878	0.936	0.929
	Precision	0.597	0.604	0.627	0.599	0.925	0.854	0.972	0.929
	Recall	0.600	0.592	0.618	0.579	0.905	0.904	0.905	0.929
RF	F1-Score	0.609	0.640	0.663	0.609	0.936	0.963	0.962	0.974
	Precision	0.788	0.892	0.932	0.855	0.972	1.000	1.000	1.000
	Recall	0.567	0.585	0.601	0.567	0.905	0.929	0.929	0.952
Gaussian NB	F1-Score	0.499	0.506	0.505	0.502	0.514	0.516	0.515	0.509
	Precision	0.503	0.505	0.503	0.504	0.510	0.509	0.510	0.507
	Recall	0.528	0.533	0.532	0.533	0.993	0.968	0.992	0.993
k-NN	F1-Score	0.564	0.594	0.598	0.601	0.880	0.888	0.879	0.880
	Precision	0.546	0.570	0.575	0.573	0.828	0.840	0.796	0.828
	Recall	0.608	0.648	0.645	0.666	0.952	0.952	0.952	0.952
LR	F1-Score	0.463	0.460	0.461	0.465	0.518	0.588	0.536	0.528
	Precision	0.504	0.506	0.506	0.506	0.511	0.549	0.518	0.516
	Recall	0.736	0.801	0.811	0.820	0.970	0.951	0.948	0.947
XGB	F1-Score	0.561	0.613	0.661	0.662	0.952	0.963	0.963	0.952
	Precision	0.537	0.573	0.610	0.621	0.952	1.000	1.000	0.953
	Recall	0.750	0.777	0.805	0.749	0.905	0.929	0.929	0.952

Table B.15: CERT r4.2 Day ICA Oversampled results

B.4.4 CERT r4.2 Day GP Oversampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.520	0.580	0.536	0.541
	Precision	0.515	0.549	0.520	0.522
	Recall	0.675	0.742	0.902	0.926
RF	F1-Score	0.520	0.582	0.532	0.541
	Precision	0.515	0.550	0.518	0.522
	Recall	0.672	0.750	0.854	0.926
Gaussian NB	F1-Score	0.492	0.498	0.401	0.750
	Precision	0.509	0.507	0.500	0.673
	Recall	0.770	0.660	0.548	0.952
k-NN	F1-Score	0.526	0.682	0.870	0.925
	Precision	0.525	0.657	0.840	0.947
	Recall	0.527	0.716	0.905	0.905
LR	F1-Score	0.493	0.486	0.510	0.544
	Precision	0.509	0.509	0.508	0.524
	Recall	0.783	0.829	0.942	0.950
XGB	F1-Score	0.511	0.552	0.513	0.527
	Precision	0.514	0.533	0.509	0.515
	Recall	0.768	0.920	0.921	0.924

Table B.16: CERT r4.2 Day GP Oversampled results

B.5 CERT r4.2 Session Oversampled

B.5.1 CERT r4.2 Session Original with Oversampling

ML Model	0vsAll			0vs1_3		
	F1-Score	Precision	Recall	F1-Score	Precision	Recall
DT	0.848	0.875	0.825	0.910	0.921	0.900
RF	0.831	0.978	0.753	0.912	1.000	0.850
Gaussian NB	0.397	0.502	0.737	0.468	0.501	0.837
k-NN	0.503	0.502	0.506	0.534	0.526	0.550
LR	0.454	0.504	0.764	0.468	0.501	0.837
XGB	0.968	0.950	0.986	0.987	1.000	0.975

Table B.17: CERT r4.2 Session Original with Oversampling

B.5.2 CERT r4.2 Day PCA Session Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		P-5	P-10	P-15	P-20	P-5	P-10	P-15	P-20
DT	F1-Score	0.584	0.613	0.592	0.642	0.868	0.917	0.878	0.889
	Precision	0.587	0.634	0.599	0.658	0.889	0.969	0.912	0.937
	Recall	0.582	0.598	0.586	0.629	0.850	0.875	0.850	0.850
RF	F1-Score	0.594	0.608	0.603	0.637	0.929	0.929	0.929	0.912
	Precision	0.797	0.782	0.854	0.893	1.000	1.000	1.000	1.000
	Recall	0.556	0.567	0.561	0.583	0.875	0.875	0.875	0.850
Gaussian NB	F1-Score	0.501	0.504	0.499	0.496	0.603	0.519	0.522	0.527
	Precision	0.502	0.504	0.503	0.503	0.561	0.511	0.512	0.515
	Recall	0.512	0.519	0.528	0.558	0.824	0.822	0.822	0.823
k-NN	F1-Score	0.569	0.622	0.607	0.644	0.921	0.910	0.936	0.932
	Precision	0.548	0.588	0.575	0.604	0.944	0.921	0.947	0.971
	Recall	0.629	0.704	0.691	0.738	0.900	0.900	0.925	0.900
LR	F1-Score	0.451	0.465	0.466	0.472	0.487	0.523	0.551	0.566
	Precision	0.504	0.506	0.506	0.506	0.502	0.513	0.528	0.536
	Recall	0.744	0.825	0.838	0.853	0.944	0.971	0.973	0.974
XGB	F1-Score	0.540	0.590	0.604	0.645	0.929	0.929	0.959	0.974
	Precision	0.525	0.570	0.565	0.599	1.000	0.970	1.000	0.999
	Recall	0.798	0.812	0.774	0.782	0.875	0.875	0.925	0.950

Table B.18: CERT r4.2 Day PCA Session Oversampled results

B.5.3 CERT r4.2 Day ICA Session Oversampled

ML Model	Metric	0vsAll				0vs1_3			
		I-5	I-10	I-15	I-20	I-5	I-10	I-15	I-20
DT	F1-Score	0.572	0.633	0.617	0.644	0.917	0.929	0.894	0.912
	Precision	0.576	0.643	0.626	0.674	0.969	1.000	1.000	1.000
	Recall	0.569	0.625	0.609	0.623	0.875	0.875	0.825	0.850
RF	F1-Score	0.597	0.646	0.639	0.651	0.929	0.912	0.894	0.912
	Precision	0.801	0.832	0.841	0.881	1.000	0.999	1.000	1.000
	Recall	0.558	0.594	0.587	0.594	0.875	0.850	0.825	0.850
Gaussian NB	F1-Score	0.505	0.506	0.500	0.497	0.509	0.514	0.524	0.510
	Precision	0.504	0.505	0.502	0.502	0.506	0.508	0.514	0.507
	Recall	0.521	0.532	0.520	0.524	0.819	0.820	0.822	0.820
k-NN	F1-Score	0.571	0.620	0.603	0.635	0.921	0.932	0.932	0.944
	Precision	0.550	0.586	0.572	0.621	0.944	0.970	0.971	1.000
	Recall	0.626	0.698	0.688	0.714	0.900	0.900	0.900	0.900
LR	F1-Score	0.471	0.464	0.467	0.472	0.487	0.517	0.561	0.575
	Precision	0.505	0.506	0.506	0.507	0.502	0.510	0.519	0.541
	Recall	0.737	0.801	0.812	0.821	0.944	0.970	0.974	0.974
XGB	F1-Score	0.548	0.611	0.638	0.734	0.917	0.962	0.974	0.974
	Precision	0.529	0.570	0.590	0.678	0.969	1.000	1.000	1.000
	Recall	0.819	0.787	0.804	0.842	0.875	0.950	0.950	0.950

Table B.19: CERT r4.2 Day ICA Session Oversampled results

B.5.4 CERT r4.2 Day GP Session Oversampled

ML Model	Metric	0vsAll		0vs1_3	
		GP-FS	GP-NFS	GP-FS	GP-NFS
DT	F1-Score	0.505	0.506	0.913	0.870
	Precision	0.507	0.511	0.865	0.827
	Recall	0.591	0.768	0.975	0.925
RF	F1-Score	0.505	0.506	0.913	0.870
	Precision	0.507	0.511	0.865	0.827
	Recall	0.591	0.768	0.975	0.925
Gaussian NB	F1-Score	0.485	0.506	0.522	0.472
	Precision	0.506	0.509	0.512	0.501
	Recall	0.751	0.696	0.822	0.919
k-NN	F1-Score	0.525	0.630	0.871	0.833
	Precision	0.519	0.616	0.933	0.875
	Recall	0.536	0.647	0.825	0.800
LR	F1-Score	0.490	0.453	0.713	0.719
	Precision	0.508	0.505	0.638	0.645
	Recall	0.813	0.824	0.975	0.950
XGB	F1-Score	0.507	0.498	0.873	0.860
	Precision	0.511	0.511	0.806	0.800
	Recall	0.748	0.913	0.975	0.950

Table B.20: CERT r4.2 Day GP Session Oversampled results