



Nottingham Trent
University

**Automated Detection of Insider Threats in Security
Operations Centres Using SIEM Data and Machine
Learning Techniques**

by

Karunakar Reddy Machupalli

in

2025

Project report in part fulfilment
of the requirements for the degree of
Master of Science
In
Cyber Security

I hereby declare that I am the sole author of this report. I authorize the Nottingham Trent University to lend this report to other institutions or individuals for the purpose of scholarly research.

I also authorize the Nottingham Trent University to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Karunakar Reddy Machupalli

ABSTRACT

Insider threats make up an increasing and complex cybersecurity challenge for modern enterprises, as they leverage the inherent trust granted to internal personnel and frequently evade detection by conventional security protocols. This project outlines the creation and execution of an automated system for detecting insider threats in Security Operations Centres (SOCs) by utilizing Security Information and Event Management (SIEM) data combined with machine learning methodologies. The study examines significant shortcomings in existing insider threat detection systems, namely the issue of "alert fatigue" encountered by security analysts because of the numerous false positives produced by traditional SIEM systems. The proposed system employs behavioural anomaly detection techniques to examine user activity patterns and detect potentially harmful activities within organizational settings. The methodology utilizes strong synthetic datasets that simulate realistic organizational circumstances, using multiple data sources like user authentication activities, device interactions, network connections, and file access patterns. Machine learning algorithms are designed to identify anomalies in typical user behaviour that may indicate insider threat activities, involving both hostile insiders and pretenders. This framework's major unique aspect is the incorporation of explainable artificial intelligence approaches, which offer understandable insights into anomaly detection judgments, enabling SOC analysts to comprehend and evaluate the explanations behind threat identifications. This method retains human supervision in decision-making while improving the efficiency and precision of threat identification. The framework aims to assist, not replace human analysts by delivering actionable intelligence that facilitates better prioritizing of security warnings. This research expands responsible machine learning applications in cybersecurity by prioritizing interpretability, scalability, and ethical considerations, providing organizations with a practical approach to solidifying defences against insider threats while maintaining operational efficiency.

ACKNOWLEDGEMENTS

I wish to convey my deep gratitude to my supervisor, Doratha Vinkemeier, for her crucial guidance and assistance during this research. I convey my thanks to the Singapore University of Technology and Design Cybersecurity Lab for granting access to the TWOS dataset, and to the CERT Division of the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) for supplying the CERT dataset. I express my appreciation to all individuals who participated in and assisted the survey, as their efforts were vital for my research.

TABLE OF CONTENTS

ABSTRACT	II
ACKNOWLEDGEMENTS	III
TABLE OF CONTENTS.....	IV
LIST OF FIGURES	VIII
LIST OF TABLES	IX
CHAPTER 1	10
INTRODUCTION	10
1.1 Background and Context.....	10
1.2 Problem Statement & Importance of the Study	12
1.3 Project Aim	14
1.4 SMART Objectives	17
1.5 Scope and Limitations.....	20
1.6 Overview of the Report Structure.....	22
CHAPTER 2	25
LITERATURE REVIEW	25
2.1 Introduction.....	25
2.2 Insider Threats in Cybersecurity	28
2.3 Behavioural Analysis for Threat Detection	29
2.4 SIEM Systems: Capabilities and Gaps.....	30
2.5 Machine Learning in Insider Threat Detection	31

2.6 Use of Datasets	32
2.7 Research Gap	33
2.8 Research Questions	34
2.9 Summary	34
CHAPTER 3	36
NEW IDEAS & APPROACH	36
3.1 Introduction	36
3.2 Rationale for Chosen Approach	37
3.3 Dataset Selection and Description	38
3.4 Reddit Survey	40
3.5 Machine Learning Model Justification	41
3.6 Overall Methodology	44
3.7 Ethical, Social, and Legal Considerations	46
3.8 Summary	47
CHAPTER 4	48
IMPLEMENTATION	48
4.1 Introduction	48
4.2 Tools and Technologies Used	49
4.3 Data Processing	50
4.4 Model Development	56

4.6 Challenges Encountered	61
4.7 Summary	62
CHAPTER 5	64
EVALUATION	64
5.1 Introduction	64
5.2 Evaluation Strategy	64
5.3 Performance Metrics Used	65
5.4 Experimental Results	67
5.5 Explainability Using SHAP	71
5.6 Interpretation of Results	72
5.7 Validation / Verification	74
5.9 Summary	76
CHAPTER 6	77
CONCLUSIONS / FUTURE WORK	77
6.1 Introduction	77
6.2 Summary of Findings	77
6.3 Achievement of Aims and Objectives	79
6.4 Reflections on Methodology and Results	81
6.5 Project Limitations	81
6.6 Recommendations for Future Work	82

6.7 Closing Remarks	83
REFERENCES	84
APPENDIX A – ACQUIRING TWOS DATASET	87
APPENDIX B - REDDIT SURVEY.....	88
APPENDIX C - DATA PROCESSING & FEATURE ENGINEERING	91
APPENDIX D - MODEL IMPLEMENTATION.....	95
MEMORY AND PROCESSING CONFIGURATION	107
MODEL CONFIGURATION	107
SHAP CONFIGURATION.....	107
FILE PATHS	107
LOGGING CONFIGURATION USED THROUGHOUT THE PROJECT	108
APPENDIX E - EXTENDED EVALUATION OF RESULTS	109

LIST OF FIGURES

Figure 1: SMART Objectives

Figure 2: Gantt chart

LIST OF TABLES

Table 1: Labelled data metrics

Table 2: Unlabelled data metrics

Table 3: CERT Labelled Dataset Performance Matrix

Table 4: TWOS Unlabelled Dataset Performance Matrix

Table 5: Combined Detection Performance and Anomaly Stability Across Contamination Rates

CHAPTER 1

INTRODUCTION

1.1 Background and Context

The tremendous digital transformation of enterprises has brought about remarkable benefits and significant cybersecurity challenges. Insider threats, including both intentional and negligent actions by workforce within a company, have surfaced as one of the most complex and damaging forms of cyber risk. Insider threats are especially dangerous as they leverage the fundamental trust offered to employees, contractors, and other internal stakeholders, making them challenge to identify through standard cybersecurity procedures. Insider threats present in several ways, such as intellectual property theft, sabotage, fraud, and unlawful access to confidential information. Notable incidents, such as Edward Snowden's disclosure of classified NSA information (*Gelles, 2016*), have highlighted the serious impact of insider threats on corporate security, reputation, and profitability. The 2023 Ponemon Institute report indicates a 44% rise in insider threats over the past two years, with the average cost of an insider-related incident totalling \$15.38 million (*Ponemon Institute, 2023*). Conventional cybersecurity tools, such as firewalls and intrusion detection systems, are mostly intended to protect against external threats and frequently neglect the complex and context-dependent traits that define insider threats (*Probst et al., 2010*). This has resulted in an increasing interest in utilizing modern technologies, such as machine learning and behavioural analytics, to identify and mitigate insider risks. Behavioural analysis has gained prominence by concentrating on the identification of anomalies in typical user behaviour

patterns, which might act as early signs of malicious intent (*Eberle et al., 2010*).

SIEM is short for Security Information and Event Management. It is an integrated cybersecurity system that integrates Security Information Management (SIM) and Security Event Management (SEM) operations. SIEM systems gather, organize, and evaluate security data from various places inside an organization's IT infrastructure, incorporating firewalls, servers, endpoints, applications, and network devices. They offer real-time surveillance of security incidents, employ correlation rules and algorithms to detect possible threats, create warnings for anomalous behaviors, and assist with incident response and compliance documentation. However, SIEM systems frequently run into issues, including alert fatigue because of high false positive rates, complex configuration requirements, and scalability constraints associated with huge data sizes.

Security Information and Event Management (SIEM) systems have emerged as fundamental components of modern cybersecurity tactics, including centralized logging and real-time surveillance of security incidents. Nonetheless, whereas SIEM systems excel at data aggregation, they frequently lack the analytical proficiency to identify complex insider threats (*Chuvakin et al., 2010*). To rectify this deficiency, researchers have suggested the incorporation of machine learning models into SIEM systems to improve their capacity for detecting aberrant behaviour. The CERT Insider Threat Center has created datasets specifically to replicate insider threat scenarios, allowing researchers to train and assess machine learning models (*Glasser & Lindauer, 2013*). Machine learning methodologies, like Isolation Forests and anomaly detection algorithms, have demonstrated efficiency in identifying insider threats through the analysis of large behavioural data (*Liu et al., 2008*). These models can detect anomalies in user behaviour, such as unusual login times or unauthorized file access, which may suggest malicious intent. Nonetheless, the execution of such models presents challenges. Challenges include data privacy, ethical concerns, and the necessity for high-quality labelled datasets remain as significant barrier to wider

use (*Maloof & Stephens, 2007*). This study seeks to tackle these difficulties by investigating the utilization of machine learning methodologies for insider threat detection. This project aims to establish a comprehensive technique for detecting and mitigating insider threats using datasets like the CERT Insider Threat Dataset and TWOS. This study's findings will build on the existing knowledge in cybersecurity and offer practical insights for organizations aiming to improve their security posture.

1.2 Problem Statement & Importance of the Study

Insider attacks constitute a significant challenge in cybersecurity, since they leverage the fundamental trust granted to employees, contractors, and other internal stakeholders. Despite breakthroughs in detection technology, firms remain encountering substantial risks owing to the volume and complexity of modern systems. Such occurrences frequently lead to substantial financial losses, reputational harm, and the compromising of sensitive information. Current solutions, SIEM systems, although efficient at collecting and aggregating security events from many sources, are fundamentally intended to identify external threats and recognized attack patterns using signature-based detection and established correlation rules. Insider threats display complex behavioural anomalies that differ from standard patterns instead of causing obvious incidents of security, making them challenging to identify through conventional rule-based methods. The complexity of insider threats arises from their contextual and behavioural characteristics, authorized users engaging in seemingly routine behaviors but, when examined cumulatively over time, may suggest malicious intent. SIEM systems have challenges with temporal and behavioural correlation, frequently producing an abundance of false positives while seeking to detect intricate patterns, resulting in alert fatigue for security analysts. Moreover, traditional anomaly detection methods generally depend on statistical thresholds

and limited deviation metrics, which inadequately identify the complex behavioural patterns associated with insider threats, including gradual changes in access patterns, complex data exfiltration activities, or the misuse of legitimate privileges for illegal activities. Moreover, these systems lack the contextual comprehension and clarity required for security analysts to differentiate between legitimate business operations and potential insider threats, especially in dynamic organizational settings where user roles and responsibilities frequently evolve. The increasing diversity and magnitude of modern company data overwhelm conventional detection methods, complicating the identification of significant anomalies while preserving acceptable standards for efficiency. SIEM systems, although proficient in aggregating and monitoring security events, often produce an excessive volume of notifications, resulting in "alert fatigue" among security teams (*Sharma et al., 2022*). This behaviour affects analysts capacity to detect and address key risks, leaving businesses vulnerable to possible breaches. Moreover, the volume of data produced by contemporary businesses complicates the identification of complex behavioural anomalies that signify insider threats (*Khan et al., 2023*). The constantly changing landscape of technology and cyber threats intensifies the issue. As firms embrace new technologies and broaden their digital presence, current detection techniques frequently fail to keep up. Machine learning algorithms have demonstrated potential in tackling these difficulties by examining extensive behavioural data and detecting anomalies from established standards. Nonetheless, these models encounter considerable challenges, such as the necessity for high-quality labelled information, ethical issues, and difficulties in integration with current systems (*Zhang et al., 2023*). Regardless these breakthroughs, vulnerabilities persist due to the magnitude and difficulty of insider threat detection. The absence of flexible, scalable, and efficient solutions leaves organizations unprepared to address the evolving nature of insider threats. This research seeks to fill these gaps by utilizing machine learning

approaches, such as Isolation Forests, and behavioural analysis to improve insider threat identification. This study aims to enhance the development of more effective and scalable solutions for insider threat mitigation by emphasizing on minimizing alert fatigue, enhancing detection accuracy, and addressing ethical considerations. This underlines the critical necessity for creative solutions to tackle this increasing issue. This study is important as it aims to fill essential gaps in current insider threat detection systems. Modern solutions, including SIEM systems, frequently produce an excessive volume of notifications, resulting in "alert fatigue" among security personnel (*Sharma et al., 2022*). Moreover, the study expands the field of cybersecurity by tackling issues associated with scalability and the evolving traits that define insider threats. The utilization of publicly accessible datasets, such as CERT and TWOS, guarantees that the proposed methodology is both pragmatic and replicable. By addressing these research gaps, the study enhances academic understanding and offers practical insights for firms aiming at improving their defences against insider threats. The finding has significant implications for the cybersecurity industry beyond its technical contributions. The study enhances the identification of insider threats, enabling firms to safeguard sensitive data, preserve confidence among customers, and prevent financial losses. Moreover, the results may direct the development of ethical and privacy-aware strategies for insider threat detection, promoting a safer and more secure digital landscape for all parties involved.

1.3 Project Aim

This project aims to develop and implement a machine learning framework for detecting insider threats in organizational settings, utilizing behavioural anomaly detection methods, particularly Isolation Forest. These algorithms will be applied on synthetic corporate logs, which incorporate those obtained from the CERT and TWOS datasets, to detect anomalous behavioural patterns that may indicate

potential insider threats. The project aims to support Security Operations Center (SOC) analysts by delivering actionable insights about unusual activity, rather than replacing human decision-making. A vital aspect of this research is the incorporation of SHAP (SHapley Additive exPlanations) in explaining the outcomes of the anomaly detection models. SHAP delivers interpretable insights into the reasoning behind the identification of specific behaviours as abnormal, providing SOC analysts with the necessary context to make informed decisions. This ensures that analysts maintain full control over the decision-making process, addressing a critical issue in cybersecurity: the transparency of machine learning models. The transparent nature of numerous modern machine learning algorithms has caused significant issues in cybersecurity contexts, where explaining the reasons behind security decisions is essential for effective incident response and forensic analysis. Security analysts must articulate and clarify their activities to stakeholders, adhere to regulatory mandates, and derive insights from both successful detections and false positives to enhance their security posture continuously. This project intentionally avoids generative or autonomous decision-making, unlike many current systems that aim to automate responses, such as quarantining endpoints, limiting user accounts, restricting user behaviors, or automatically isolating network segments. While automated answers may appear efficient, they pose considerable risks in organizational settings, as false positives can interrupt essential business operations, reduce employee productivity, and potentially breach privacy laws or employment regulations. Autonomous systems may lack the contextual comprehension required to differentiate between genuine urgent business activity and true hostile actions, especially in scenarios requiring executives, emergencies, or time-critical corporate processes. Moreover, the intricacy of insider threats often requires human insight to accurately evaluate the severity, intent, and suitable response level. In contrast to external cyberattacks that could prompt automated control, insider threat situations often require complex

considerations, including employee relations, legal consequences, evidence preservation for possible prosecution, and the necessity to balance implementing safety measures with maintaining a productive workplace. By ensuring human oversight, the framework guarantees that security responses are proportionate, legally compliant, and consistent with business goals, while also maintaining the trust and confidence required for an effective workplace security culture. The framework is intended to identify and explain abnormal activity, providing analysts with the means of prioritizing warnings more efficiently. This method directly tackles the problem of "alert fatigue," when analysts become overwhelmed by the excessive number of alerts produced by conventional Security Information and Event Management (SIEM) systems. The framework seeks to improve the efficiency and precision of threat identification by minimizing noise and offering clear explanations, while avoiding further complexity or automation concerns. The primary objective of this initiative is to assist SOC (Security Operations Center) analysts in enhancing their capacity to identify and address internal threats. The framework seeks to bring together powerful machine learning methodologies with practical, user-friendly cybersecurity technologies by emphasizing interpretability, scalability, and human oversight. This research extends the subject of cybersecurity by demonstrating the responsible and ethical application of machine learning to improve human decision-making rather than replacing it.

1.4 SMART Objectives

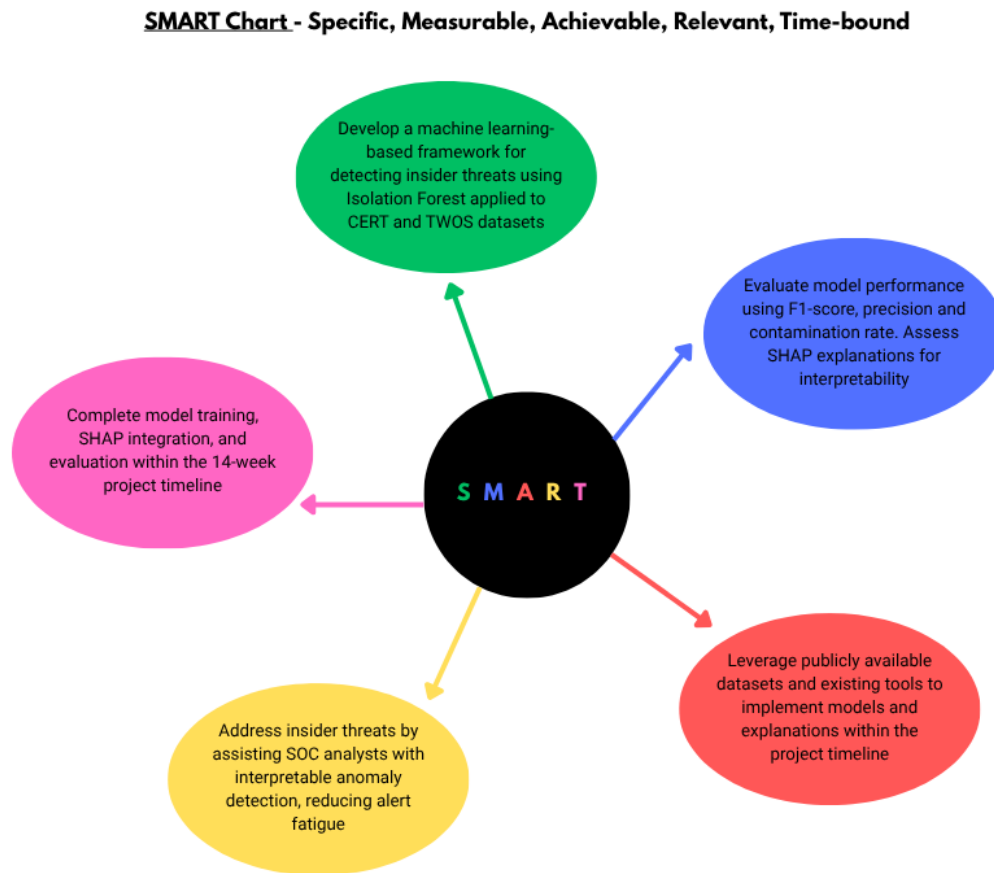


Figure 1: SMART Objectives

1.4.1 Specific

The project aims to establish a machine learning framework for identifying insider threats through behavioural anomaly detection methods, particularly utilizing Isolation Forest. The algorithms will be applied on synthetic corporate logs, including the CERT and TWOS datasets, to detect anomalous behavioural patterns suggestive of insider threats. The architecture aims to aid SOC analysts by highlighting and explaining unusual activity, preventing autonomous decision-making, and guaranteeing human oversight.

1.4.2 Measurable

The project's success will be assessed using clearly established metrics. The effectiveness of the anomaly detection models will be evaluated using F1-score, precision, and contamination rate. The outcomes' understanding will be evaluated by SHAP explanations, guaranteeing that the identified anomalies are comprehensible and actionable for SOC analysts. The framework's capacity to mitigate "alert fatigue" will be demonstrated by comparing the number of false positives produced.

1.4.3 Achievable

The project is structured to be practical and attainable within the specified limitations. Publicly accessible datasets, including CERT and TWOS, will be utilized for the training and evaluation of the models, hence ensuring accessibility and repeatability. Existing libraries and tools for implementing Isolation Forest and SHAP will be utilized to facilitate development and focus on the primary objectives. By avoiding complex autonomous decision-making systems, the project scope stays feasible while effectively tackling essential issues in insider threat detection.

1.4.4 Relevant

The project addresses a critical challenge in cybersecurity the identification of insider threats, which are notoriously challenging to detect and manage. The framework enhances alert fatigue reduction and triage efficiency by equipping SOC analysts with tools for identifying and analysing unusual behaviour. This corresponds with the overall objective of improving human decision-making in cybersecurity instead than replacing it with automated solutions. The study further advances the field by illustrating the responsible application of machine learning to address real-world issues.

1.4.5 Time-bound

The project is structured with a clear timeline to ensure timely completion:

The project is organized over a 14-week period, beginning with the proposal and supervisor approval in the initial two weeks, succeeded by background research and an extensive literature evaluation to construct the study's basis. During weeks 3 and 4, the focus transitions to the identification and prepping of datasets, ensuring that the synthetic enterprise logs (CERT and TWOS datasets) are sanitized, structured, and prepared for model development. In weeks 5 and 6, machine learning methods such as Isolation Forest are chosen, and baseline models are established, resulting in the creation of a preliminary operational model for anomaly detection. Weeks 7 and 8 focus on model tuning and feature engineering, optimizing speed, and including SHAP explanations to improve interpretability for SOC analysts. The framework's evaluation and validation occur in weeks 9 and 10, utilizing measures including F1-score, precision and contamination rate. The results have been finalized, and professional issues, including ethical and practical considerations, have been recorded. The final report composition commences in weeks 11 and 12, encompassing all chapters and conclusions, with the draft submitted for evaluation and critique. The final two weeks, 13 and 14, are designated for adjusting based on feedback, completing the report, and submitting the major project for evaluation. This organized plan guarantees methodical advancement and punctual fulfilment of all project milestones.

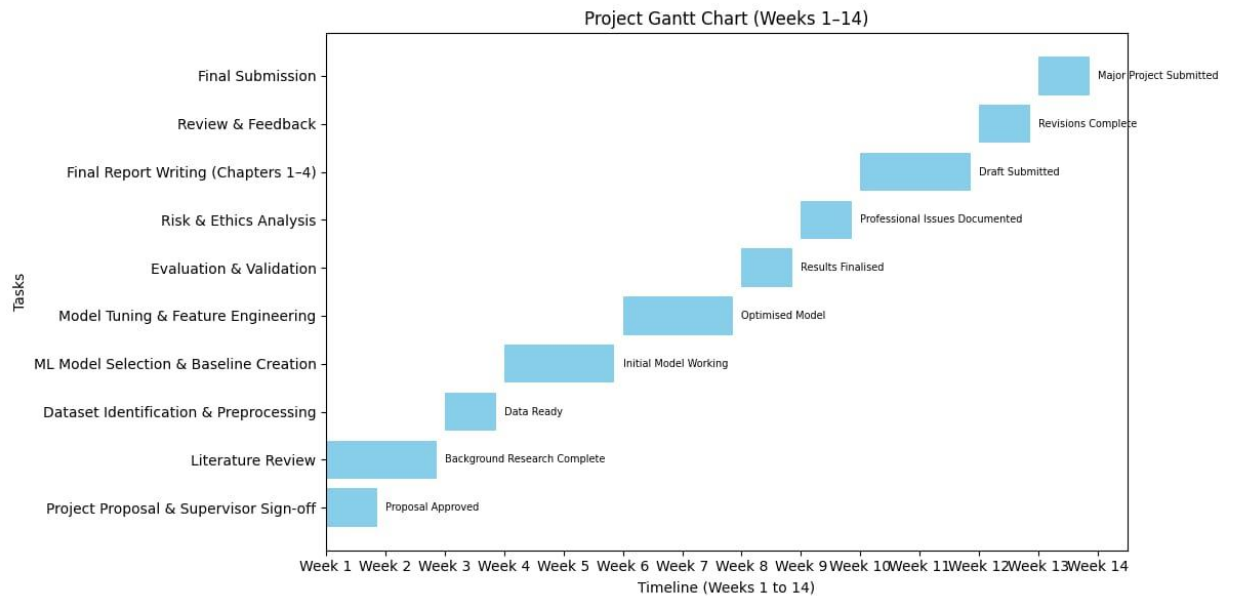


Figure 2: Gantt chart

1.5 Scope and Limitations

1.5.1 Scope

This study examines the application of machine learning methodologies for identifying insider threats inside organizational settings, focusing behavioural anomaly detection. The research uses the Isolation Forest model applied to synthetic enterprise logs derived from the CERT and TWOS datasets. The major goal is to assist Security Operations Center (SOC) analysts by highlighting and explaining anomalous activity using SHAP (SHapley Additive exPlanations), thus ensuring clarity and retaining human oversight. This approach, in contrast to numerous existing solutions, refrains from autonomous decision-making, such as quarantining endpoints or obstructing user operations, and prioritizes delivering actionable insights to improve triage efficiency. The study's scope includes the processing of two different types of data obtained from the raw datasets:

Labelled Data: This subset features predetermined labels that facilitate the quantitative evaluation of machine learning models through metrics such as F1-

score, precision, and recall. The labelled data underpins the validation of the anomaly detection framework's accuracy and dependability.

Unlabelled Data: This subset has no presence of predetermined labels, making it inappropriate for assessment by conventional metrics such as F1-score. The unlabelled data is analysed to detect behavioural anomalies, which are then flagged for further examination by SOC analysts. This method emphasizes the framework's interpretability and practical utility in real-world contexts.

The outputs of this research encompass: A functional framework for anomaly detection that can handle both labelled and unlabelled data. Utilizing SHAP for interpretable data to furnish SOC analysts with actionable insights into identified anomalies. An in-depth examination of the framework's capacity to mitigate alert fatigue and enhance triage efficiency in Security Operations Center environments. This research enhances the field of cybersecurity by tackling the complexities of insider threat detection, emphasizing clarity, scalability, and ethical implications.

1.5.2 limitations

Dataset Limitations: The research used synthetic datasets (CERT and TWOS), which, although widely used in academic studies, may not entirely capture the intricacies, diversity, and unpredictability of actual industrial contexts. This constraint could impact the applicability of the findings to practical environments.

Evaluation Scope: The framework's efficacy may only be quantitatively assessed on labelled data with metrics such as F1-score, precision, and recall. The assessment of unlabelled data is confined to qualitative analysis, depending on the comprehension of SHAP explanations and the practical implications of identified anomalies for SOC analysts. The research is confined to anomaly

detection and understanding, deliberately excluding autonomous actions like quarantining endpoints or obstructing user activity. This guarantees human supervision but restricts the framework's capacity to aid analysts instead of automating replies, which could be perceived as a disadvantage in situations necessitating swift action.

Scalability and Resource Limitations: The system is optimized for batch processing and may encounter difficulties when used to larger datasets or real-time contexts. Further efforts will be necessary to resolve these scaling issues and modify the architecture for dynamic enterprise systems. The research is confined to a 14-week period, which restricts the extent of experimentation, validation, and scalability assessment. Resource constraints further inhibit the capacity to investigate alternate datasets, models, or deployment scenarios. The study's contributions aim to establish a basis for future research in this field, especially concerning scalability and the integration of real-world datasets.

1.6 Overview of the Report Structure

The report consists of six chapters, each focusing on an individual aspect of the research and enhancing the extensive research into insider threat identification through behavioural anomaly detection methodologies. The framework is intended to facilitate an orderly advancement from the presentation of the study issue to the execution, assessment, and conclusions.

Chapter One: Introduction

The initial chapter lays the groundwork for the investigation by presenting the subject of insider threat detection and its importance in cybersecurity. It presents the study's background and context, establishes the problem statement, and clarifies the significance of the research. The chapter establishes

the project aim, SMART objectives, scope, and limitations, so ensuring clarity regarding the focus and boundaries of the research.

Chapter Two: Literature Review

The second chapter examines current research and advancements in insider threat detection. It examines important issues like insider threats in cybersecurity, behavioural analysis for threat identification, the strengths and weaknesses of SIEM systems, and the implementation of machine learning methodologies, such as the Isolation Forest model. This chapter examines the use of datasets such as CERT and TWOS, highlighting deficiencies in the existing body of knowledge that the research intends to address.

Chapter Three: New Ideas and Approach

The third chapter defines the methodology and justification for the selected approach. The content outlines the selection and preprocessing of datasets, the justification for employing Isolation Forest and datasets, and the incorporation of SHAP explanations to improve clarity. The chapter includes discussions on ethical, social, and legal aspects, guaranteeing that the research complies with responsible practices.

Chapter Four: Implementation

The fourth chapter outlines the technical execution of the framework. It encompasses the preprocessing of both labelled and unlabelled data, feature extraction, and the training of machine learning models. The challenges faced during implementation are examined, offering insights into the practical dimensions of the research.

Chapter Five: Evaluation

Chapter five assesses the framework's performance using criteria like F1-score, precision, recall, and contamination rate for data. The meaning of SHAP explanations for unlabelled data is evaluated to determine their effectiveness in aiding SOC analysts. This chapter evaluates outcomes, evaluates findings, and confirms the framework's efficacy in mitigating alert fatigue and enhancing triage efficiency.

Chapter Six: Conclusions and Future Work

The concluding chapter summarizes the research findings, discussing the approach and outcomes. The report addresses the study's shortcomings and offers recommendations for future research, addressing scalability, integration of real-world datasets, and ethical considerations. The chapter finishes with final observations regarding the research's contributions to the field of cybersecurity.

Appendices

Supplementary materials, including screenshots, logs, and supplementary data, are incorporated in the appendices to support the key elements of the report.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Insider threats emerge when individuals within an organization "employees, contractors, or business partners" take advantage of their permitted access to inflict harm on the organization (*Homoliak et al. 2019*). In comparison to external hackers who aim to infiltrate systems from the outside, insiders have legitimate access to organizational data and networks, which makes their malicious behaviour significantly more challenging to identify. Notable cases involve Edward Snowden's disclosure of classified NSA documents in 2013, a Capital One employee's unauthorized access to 100 million customer records in 2019, and Tesla employees who illegally sold consumer data to rival companies. These instances highlight how trustworthy insiders may cause serious damage on companies. The magnitude of this issue is increasing promptly and poses a considerable risk to business. Recent studies indicate a 44% rise in insider incidents over the past two years, with each incidence imposing an average cost of \$15.38 million on enterprises (*Ponemon Institute 2023*). Most alarming is that 60% of these attacks require months or years for detection, during which the damage continues in growing. The delayed detection mostly arises from existing security systems producing excessive false alarms, resulting in "alert fatigue," whereby security analysts become overwhelmed and overlook authentic dangers under numerous routine notifications. Conventional security monitoring predominantly depends on Security Information and Event Management (SIEM) systems; however, these systems possess built-in limitations in addressing insider threats. A conventional SIEM system produces thousands of warnings

each day, the majority of which are false positives that consume analysts' time and focus. These systems have difficulty recognizing context; for instance, they may categorize an employee working late on a project deadline as suspicious activity just due to its occurrence outside standard business hours. Moreover, conventional SIEM systems utilize static rules that fail to adapt to evolving user behaviour patterns, and when alarms are triggered, they lack explanations for the classification of actions as suspicious, restricting analysts' ability to make informed decisions. This literature review analyses current research in six essential domains that contribute to the advancement of more efficient insider threat detection systems. Initially, it examines the many categories of insider threats, including hostile insiders who actively inflict harm on businesses, negligent insiders who accidentally trigger damage, and compromised insiders whose accounts have been stolen by external attackers. Secondly, it examines behavioural analytic approaches that outline typical user behaviour patterns and detect anomalous actions while reducing false positives. Third, it examines existing SIEM systems to understand their functionalities and constraints in identifying insider threats. The review subsequently analyses machine learning approaches for insider threat detection, focusing on the Isolation Forest algorithm and the advantages of unsupervised learning techniques over conventional rule-based systems in identifying previously overlooked threat patterns (*Liu, Ting, and Zhou 2008*). The evaluation includes accessible research datasets, primarily the synthetic CERT dataset from Carnegie Mellon University and the TWOS dataset with actual user behaviour data from Singapore, analysing the advantages and drawbacks of employing synthetic versus actual data for the development of detection algorithms. Finally, it examines explainable artificial intelligence methodologies, namely SHAP (SHapley Additive exPlanations) technology, that assist security analysts in understanding the reason behind the identification of certain behaviors as suspicious (*Lundberg and Lee 2017*). This literature evaluation points out three significant deficiencies in

existing research that this study seeks to address. mainly, the majority of current research perform effectively in laboratory environments but falls short when implemented in actual Security Operations Centers, where analysts are required to manage numerous alerts on a regular basis. Secondly, numerous AI-driven detection systems function as "black boxes," making their decision-making processes transparent, which complicates trust and meaningful responses for security analysts. Third, not enough study fully addresses the ethical consequences of behavioural monitoring systems, encompassing employee privacy issues and the risk of unfair automated decision-making (*Bhatt et al. 2020*). This research adopts a fundamentally distinct approach by focusing practical implementation in actual SOC contexts. Instead of substituting human analysts with automated systems, it results in tools that assist analysts in decision-making by offering clear justifications for the identification of particular behaviors as suspicious. For example, rather than essentially indicating that "User X is suspicious," the system could explain that "User X accessed files tenfold more than normal, performed at 3 AM beyond standard hours, and transferred data to a USB drive - a behaviour identical to recognized data theft incidents." This method minimizes false positives and reduces alert fatigue while guaranteeing that analysts understand and can respond to the insights generated by the detection system. This literature review's structure highlights the broad aspect of insider threat detection research, with each section refining previous evaluations to establish a thorough foundation for the proposed study approach. The review concludes by outlining specific research questions and hypotheses that direct the experimental design and procedures discussed in subsequent chapters.

2.2 Insider Threats in Cybersecurity

Insider attacks are one of cybersecurity's toughest challenges, as they involve individuals who already have authorized access to company systems and data. In comparison with external attackers who must penetrate security perimeters, insiders bypass these defences fully by utilizing their valid credentials, leaving detection very challenging with conventional security procedures (*Cappelli, Moore, and Trzeciak 2012*). Existing studies categorize insider risks into three major types. Malicious insiders intentionally exploit their access to steal information, perform fraud, or disrupt systems, often motivated by the desire for money, hatred, or ideological principles. Unintentional insiders accidentally introduce security vulnerabilities through reckless actions, such as falling victim to phishing attacks, using weak passwords, or ignoring obeying to security protocols. Compromised insiders emerge when outsiders obtain control of legitimate user accounts through credential theft or malware, helping them to carry out tasks within organizational networks while impersonating as normal users (*Homoliak et al. 2019*). The financial consequences of insider threats have escalated significantly, with events currently averaging \$15.38 million in costs per occurrence, reflecting a 44% rise in recent years (*Ponemon Institute 2023*). The most alarming part is that these occurrences usually remain undetected for months or years, during which damage keeps on accumulating. Psychological study has revealed many factors that may contribute to insider threat behaviour, including work disappointment, stress over money, and serious changes in one's life. The connection between these risk indicators and actual insider actions is complex and non-predictive, as numerous employees encounter these obstacles without posing dangers (*Shaw and Stock 2011*). Moreover, several insider attacks happen without clear prior indicators, making avoidance very difficult. Typical mitigation strategies depend on background investigations, limitations on access, security protocols, and staff surveillance. Although these measures offer

fundamental security, they have considerable limits in addressing complex insider threats. Background checks cannot predict future conduct, access controls may be avoided by authorized individuals, and monitoring leads to privacy issues (*Greitzer et al. 2014*).

2.3 Behavioural Analysis for Threat Detection

Behavioural analysis is a cybersecurity technique that monitors and analyses user behaviors to detect patterns that diverge from defined expected standards (*Legg et al. 2017*). compared to conventional security approaches that depend on predetermined rules or signatures to identify known risks, behavioural analysis is focused on understanding regular user behaviour and identifying activities that deviate from these established norms. In this context, threat detection means the identification of potentially malicious behaviors through the identification of anomalies in user behaviour that may suggest insider threats or compromised accounts. The primary principle of behavioural analysis is to develop unique behavioural baselines for each user inside an organization. These baselines outline standard patterns like login timings, file access frequencies, network utilization, email communication patterns and application usage. For example, if an employee generally operates from 9 AM to 5 PM on weekdays, accesses 20 to 30 files daily, and communicates via email with 5 to 10 coworkers, the system recognizes these patterns as standard behaviour. When a similar person unexpectedly signs in at 3 AM, views 200 files, and communicates with external contacts via email, the behavioural analysis system will identify this as unusual activity justifying additional investigation (*Brown et al. 2019*). User and Entity Behaviour Analytics (UEBA) has emerged as a unique approach to behavioural analysis within cybersecurity environments. UEBA systems merge data from several sources, including authentication logs, file access records, network traffic, and email conversations, to create detailed behavioural profiles

for people and entities inside an organization. This broad approach offers better anomaly identification by considering in context and relationships across several data sources instead of examining individual incidents (*Gavai et al. 2015*). The benefits of behavioural analysis surpass the features of standard rule-based detection systems significantly. Conventional systems produce a multitude of false positives due to their inability to comprehend context or adjust to genuine change in user behaviour. Behavioural analysis minimizes false positives by learning and adapting to individual user behaviors while ensuring responsiveness to actual threats. This method can also discover previously unidentified ways of attacking and zero-day vulnerabilities that would not activate conventional signature-based systems, making it particularly beneficial for detecting advanced insider threats that develop over time.

2.4 SIEM Systems: Capabilities and Gaps

Security Information and Event Management (SIEM) systems are centralized platforms that gather, aggregate, and analyse security event data from many sources inside an organization's IT infrastructure (*Kent and Souppaya 2016*). SIEM solutions aggregate logs from firewalls, servers, endpoints, apps, and network devices, unifying this diverse data into a uniform format for analysis. Their main functions involve real-time security surveillance, event correlation across several systems, automated notifications for suspected actions, compliance documentation, and forensic analysis assistance for investigation of incidents. Modern SIEM systems specialize in data aggregation and correlation. For example, they can link a failed login attempt from an external IP address with following firewall blocks and security system alerts to determine targeted attack patterns. They offer strong audit trails crucial for regulatory compliance and can handle millions of events every day from numerous sources. Furthermore, SIEM systems provide customisable dashboards and reporting

functionalities that assist security teams in visualizing threats and monitoring security metrics over time (*Chuvakin, Schmidt and Phillips 2013*). However, SIEM systems have significant shortcomings in mitigating insider threats. They lack contextual awareness of user roles and company processes, limiting their ability to differentiate between legitimate anomalous actions and authentic threats. For example, when a finance manager accesses payroll systems at month-end, SIEM systems are unable to distinguish this normal business activity from unauthorized insider access. They also have difficulties with time correlation over long periods of time, failing to detect gradually emerging insider threats that evolve over weeks or months instead of immediate attack sequences (*Miloslavskaya and Tolstoy 2016*). Moreover, SIEM systems require intensive manual configuration and rule maintenance. They are unable to autonomously adjust to organizational changes, including new apps, altered business processes, or changing user roles, mandating continual upgrades by security specialists. This lack of flexibility becomes more problematic in unpredictable circumstances when authentic user behaviour patterns constantly shift due to project assignments, annual financial cycles, or organizational restructuring.

2.5 Machine Learning in Insider Threat Detection

Machine learning algorithms provide automated techniques for detecting insider threats by analysing trends from historical data without the need for explicit rule programming (*Omar et al. 2013*). Unsupervised learning approaches, especially anomaly detection algorithms like Isolation Forest and One-Class SVM, are effective at detecting outliers in user behaviour without the requirement of labelled training data for potential risks. These algorithms simultaneously examine various behavioural dimensions, such as file access patterns, network connections, email conversations, and system usage, to create detailed

behavioural profiles that cannot be manually constructed. The effectiveness of machine learning is in its ability to adapt and recognition of patterns. The Isolation Forest algorithm can identify an individual who typically views 15-20 files daily but unexpectedly downloads 500 files in a single session, even if this particular case was not expressly coded as suspect behaviour. The system detects anomalies by evaluating the simplicity with which a data point can be distinguished from typical patterns, hence proving successful in identifying unique attack tactics (*Liu, Ting and Zhou 2008*). Even so, machine learning techniques encounter considerable obstacles in the detection of insider threats. Moreover, these systems encounter difficulties with explainability, considering that complex algorithms such as deep neural networks operate as black boxes, complicating the ability of security analysts to fully understand the reasoning behind the classification of certain activities as suspicious. The absence of interpretability undermines trust and complicates incident investigation procedures (*Tuor et al. 2017*).

2.6 Use of Datasets

Datasets serve the basis for the development and assessment of insider threat detection systems, enabling controlled environments for the training and validation of algorithms against established threat scenarios (*Glasser and Lindauer 2013*). The CERT Insider Threat Dataset, created by Carnegie Mellon University, is the most widely used benchmark dataset, consisting synthetic data that portrays five insider threat scenarios, including sabotage, intellectual property theft, and data exfiltration. This dataset contains logon records, file access logs, email conversations, and HTTP traffic, allowing researchers to evaluate detection techniques across various data dimensions simultaneously. The primary significance of datasets lies in their capacity to offer ground truth labels for harmful acts, which are exceedingly rare in actual workplaces. The

CERT dataset comprises 17,000 user records over 18 months, with roughly 70 verified hostile insiders. In the absence of such datasets, researchers would find it challenging to create algorithms that can differentiate between typical changes in user behaviour and authentic malicious actions (*Lindauer et al. 2020*).

Nevertheless, synthetic datasets encounter considerable constraints that impact the generalizability of research outcomes. Synthetic datasets may generate false patterns missing in authentic organizational data, which could result in algorithms underperforming in production environments (*Bridges et al. 2020*).

2.7 Research Gap

Recent research on insider threat detection reveals a notable deficiency in integrating explainable artificial intelligence (XAI) with unsupervised anomaly detection within real-time Security Information and Event Management (SIEM) systems (*Arrieta et al. 2020*). Although current research has effectively applied machine learning algorithms for threat identification, it neglects the essential requirement for interpretable outcomes that security analysts can comprehend and utilize in practical environments. Most of the research emphasizes enhancing detection accuracy while neglecting the essential need for explainability in security operations centres, where analysts are required to justify their investigative choices. The specific research gap applies to the merging of SHAP (SHapley Additive exPlanations) or LIME XAI methodologies with unsupervised learning algorithms applied to multi-dimensional SIEM data streams. For instance, though research indicates that Isolation Forest may attain 85-90% detection accuracy on benchmark datasets, none sufficiently explain how security analysts might comprehend the reasons for the simultaneous flagging of individual users as anomalous across many behavioural aspects. The absence of explainability is a practical challenge to real-world implementation,

since security teams want actionable insights instead of abstract estimations that guide their threat investigation procedures (*Ribeiro et al. 2016*).

2.8 Research Questions

The identified research gaps require specific research questions that tackle both technological and operational issues in insider threat identification within security operations centres. The main investigation addresses the gap in explainability: "How can explainable artificial intelligence methods be incorporated with unsupervised anomaly detection algorithms to yield interpretable insights for security analysts examining insider threats?" The question at hand specifically refers to the operational needs for actionable intelligence, as opposed to unclear projections that fail to inform the investigation process (*Lundberg and Lee 2017*). Supporting studies investigate the practical issues of implementation. What is the most efficient feature engineering approach to using multi-dimensional SIEM data to enhance both detection accuracy and explaining quality? tackles the technical difficulties of processing diverse security logs while ensuring interpretability. Furthermore, "What is the impact of integrating explainability frameworks on detection performance and computational efficiency in real-time security monitoring settings?" evaluates the trade-offs between interpretability and system performance that security operations centres must consider when implementing such solutions. These research problems are crucial as they shift focus from solely algorithmic enhancements to practical implementation factors (*Molnar 2020*).

2.9 Summary

This literature study has analysed the essential elements of insider threat detection in security operations centres, highlighting notable difficulties and

opportunities for improvements. The investigation reveals that although conventional SIEM systems are efficient in data gathering and compliance reporting, they are deficient in contextual comprehension and adaptability necessary for efficient insider threat detection. Behavioural analysis methodologies present potential solutions by user activity observation, although encounter limitations in addressing temporal correlations and organizational complexities. Machine learning techniques, especially unsupervised anomaly detection algorithms, provide automatic pattern recognition abilities that can discover new risks without established criteria. The insider threats with a lack of explainability, presents considerable obstacles to practical implementation. Analysis of datasets indicates that synthetic benchmarks such as CERT offer significant research foundations, they may insufficiently reflect the complexities of real-world organizations. The acknowledged research gap focuses on the integration of explainable artificial intelligence with unsupervised learning in SIEM systems, resulting in research questions that tackle both technical implementation and operational deployment issues.

CHAPTER 3

NEW IDEAS & APPROACH

3.1 Introduction

This chapter outlines the approach utilized to create an explainable insider threat detection system that overcomes the research shortcomings found in the literature review. This method integrates the automated pattern recognition features of anomaly detection algorithms with the transparency demands of operational security contexts, where analysts are required to clearly understand and justify their investigation choices (*Adadi and Berrada 2018*). This approach tackles the following significant challenges: the requirement for interpretable outputs in security operations, and the need for real-time processing of multi-dimensional SIEM data. The study combines a quantitative experimental design with established datasets, including the CERT Insider Threat Dataset and TWOS Dataset, to assure reproducibility and allow comparison with existing techniques. The approach utilizes Isolation Forest as the principal anomaly detection algorithm, owing to its effectiveness in managing high-dimensional data and its processing efficiency in real-time contexts. The incorporation of the SHAP (SHapley Additive exPlanations) architecture offers immediate explanations for identified abnormalities, allowing security analysts to understand specific behavioural characteristics that provided threat detection. For example, when the system identifies a user accessing 200 files in one session, in contrast to their typical baseline of 20 files, SHAP explanations can pinpoint the specific features (file access frequency, time of access, file types) that most significantly influenced the anomaly score, offering actionable insights for threat investigation. This approach can be justified as it addresses the gap between

academic research focusing on detection accuracy and the actual deployment needs within enterprise security scenarios. This method allows security operations centres to execute automated threat detection while preserving the interpretability required for efficient incident response and adherence to company security standards.

3.2 Rationale for Chosen Approach

The choice of Isolation Forest with SHAP explainability demonstrates an ideal balance between detection efficiency and practicality for security operations centres. The Isolation Forest algorithm was selected due to its lack of requirement for labelled training data, making it appropriate for insider threat detection where incidents of malicious behaviour are very uncommon. Additionally, it has exceptional efficiency in feature spaces with high dimensions similar to SIEM systems (*Zhou et al. 2019*). In contrast with density-related algorithms that encounter problems due to the drawback of volume, Isolation Forest exhibits stable performance across various behavioural aspects simultaneously. Labelled data means datasets in which each data point is labelled with known outcomes, such as "malicious" or "normal" user behaviour, whereas unlabelled data comprises solely the raw behavioural aspects free from threat classifications. Labelled data consists of user login records explicitly categorized as "insider threat" or "legitimate activity," while unlabelled data includes the identical login records without any threat indications. This research applies both unlabelled data training approaches and labelled data training approaches making the model ideal choice. The integration of SHAP offers subsequent explanations crucial for security analyst workflows, selected over LIME (Local Interpretable Model-agnostic Explanations) because of its superior theoretical structure and consistency guarantee. SHAP provides globally consistent feature explanations grounded in guaranteeing that feature

importance values accumulate to the gap between the model's prediction and the baseline, whereas LIME delivers only local approximations that may differ among similar instances. For example, when identifying an abnormal user accessing databases outside standard working hours, SHAP can quantify that "time of access" accounted for 40% of the anomaly score, while "database type" contributed 25%, supporting customized analysis instead of generic threat hunting. This approach seems appropriate as supervised techniques require large labelled datasets of insider threats, which are often inaccessible in many organizations.

3.3 Dataset Selection and Description

3.3.1 CERT Dataset

The CERT Insider Threat Dataset provides synthetic organizational data that includes both normal and malicious user behaviors, making it suitable as labelled data for validation and evaluation. The dataset comprises 17,000 individuals over 18 months, including 70 verified insider threats across five scenarios: intellectual property theft, sabotage, and data exfiltration operations. Each record contains behavioural characteristics such as logon times, file access patterns, email communications, and HTTP traffic, supported with explicit threat labels indicating malicious actions (*Glasser and Lindauer 2013*). The CERT dataset represents the most detailed publicly accessible benchmark for insider threats, much outperforming other options in both scope and complexity. The KDD Cup 1999 dataset comprises 494,021 records exclusively related to external network intrusions, while the DARPA dataset offers less than 1,000 user simulations over brief intervals. In contrast, the CERT dataset includes 17,000 users producing millions of behavioural records across various data dimensions over extended durations. Alternative datasets, such as LANL authentication logs, have no ground truth labels for insider threats, whereas academic datasets like

CMU-CERT are limited in scope, including fewer than 5,000 individuals and exhibiting low behavioural complexity. This dataset fulfils two functions in the approach: it trains the Isolation Forest algorithm on unlabelled normal behaviors and supplies labelled threat samples for evaluation of performance. The dataset's standardized ecosystem guarantees reproducible outcomes and enables comparison with established methods for research. The CERT dataset seems appropriate as it offers ground truth labels crucial for numerical evaluation, while preserving realistic organizational complexity, including diverse data sources, user roles, and temporal patterns that reflect enterprise security environments.

3.3.2 TWOS Dataset

The TWOS (Trustworthy Workforce for SOC) Dataset offers real organizational data sourced from real enterprise settings, making it suitable for unlabelled data training where authentic user behavioural patterns are crucial for effective anomaly identification. In contrast with synthetic datasets, TWOS comprises real user behaviors, encompassing email conversations, keystroke dynamics, mouse movements, network traffic, and personality assessments from actual employees engaged in regular corporate operations. The dataset includes many behavioural dimensions free of inaccurate threat labels, allowing the Isolation Forest method to establish true normal behavioural baselines that represent real organizational complexity (*Bowen et al. 2021*). This dataset serves as unlabelled training data, as it incorporates spontaneous changes in user behaviour without predefined threat classifications. In fact, TWOS email data records genuine communication patterns, including diverse message frequencies, recipient distributions, and temporal trends that accurately represent actual workplace dynamics rather than synthetic behaviors. The lack of threat labels makes it appropriate for unsupervised learning methods that must identify anomalies purely through deviations from recognized regular patterns. The TWOS dataset can be justified

since actual organizational data offers more precise behavioural baselines than synthetic equivalents, hence improving detection accuracy in real company settings. Access to the dataset was obtained by a formal request to corplab-project3@sutd.edu.sg, with the relevant information included in the appendix A.

3.4 Reddit Survey

A survey was carried out on the r/cybersecurity subreddit to collect professional insights on AI-generated alerts in Security Operations Centres, yielding 265 total responses from cybersecurity professionals. The survey asked, "How much likely do you trust AI-generated alerts in SOCs?" with findings indicating little trust. 54% voted "Not at all," 30% remained "Neutral," and merely 15% indicated an intention to "Fully trust them." This distribution indicates considerable mistrust among professionals over autonomous AI decision-making in security contexts. Key discussion ideas developed from 34 professional remarks, uncovering complex viewpoints that extend beyond fundamental confidence indications. Practitioners underscored that AI should be used as a complement rather than a substitute technology, with one Security Analyst stating, "use it as an intern" able to accumulate information and issue alerts but requiring human supervision for critical decisions. Industry experts acknowledged the significance of explainability, with some respondents inquiring especially about SHAP and LIME approaches for understanding AI decision-making processes within their firms. The poll indicated a substantial difference across AI applications: experts demonstrated increased confidence in AI for anomaly identification and minimizing false positives rather than in completely autonomous threat response. An Incident Responder observed that their firm effectively employs machine learning for web application security, achieving "low false positives" and improved trust levels, while complying to traditional correlation standards for compliance purposes. Professionals constantly stressed the importance of a

"trust, but verify" approach, highlighting that careful evaluation, validation, and transparency are necessary, irrespective of whether systems for detection employ AI or standard correlation reasoning. Professional's concerns focused on the quality of implementation rather than the AI technology itself, with respondents remarking that "ineffective alert logic can be written for machine learning or correlated event alerts." The survey validated the research goal of developing explainable AI outputs to assist analyst decision-making instead than replacing human expertise, hence confirming the research approach's goal of interpretable insider threat detection for security operations centres.

Documentation can be found in appendix B.

3.5 Machine Learning Model Justification

3.5.1 Isolation Forest

The Isolation Forest algorithm operates on the idea that anomalous behaviors can be clearly distinguished from typical behaviors. In the field of insider threat identification, this is particularly useful as malicious behaviors are uncommon and typically deviate from the overall pattern of user behaviour (*Liu et al., 2012*). The technique creates several random binary trees, gradually dividing the data. At every stage, a random feature is selected, such as "number of failed logins" or "amount of data transferred," followed by the application of a random split value. The reason for this is that anomalous data points will be isolated in fewer steps than typical ones. If most employees transmit between 100–500 MB of data during working hours, a user transferring 10 GB of data at midnight will rapidly be distinguished in only a few segments. On the other hand, a user transferring 200 MB at 3 PM will necessitate numerous further splits due to their behaviour aligning with everyone else. This method enables the algorithm to identify anomalies effectively without depending on distance measurements or density evaluations, which are highly computational in high-dimensional datasets

like SIEM logs with numerous features. The algorithm's efficiency is one of its most significant advantages. The Isolation Forest algorithm has linear time complexity, $O(n)$, and maintains constant memory requirements, allowing it to efficiently process huge data volumes in real time. An organization analysing 10,000 user behaviors daily across 50 behavioural variables can employ Isolation Forest to swiftly identify the top 1% of most suspicious users, in contrast to traditional clustering or density-based methods that may need hours. The output is an isolation score that signifies the ease with which a data point can be distinguished from typical patterns. This score ranks suspicious behaviors, assisting security analysts in prioritizing alarms for investigation. Another advantage of Isolation Forest is its built-in capacity to capture complex relationships among characteristics due to its random partitioning mechanism. Unlike models requiring extensive feature engineering, it can detect anomalous patterns arising from behavioural combinations. For example, logging in during atypical hours may not automatically raise suspicion, nor may the transfer of important information. However, when these behaviors occur simultaneously, they may signify malicious intent, which Isolation Forest is more effective at detecting through its random splitting mechanism (*Hariri et al., 2019*).

3.5.2 Contamination Rate Concept

The contamination rate is a crucial metric in Isolation Forest that specifies the expected number of anomalous incidents in a dataset. This concept is essential for understanding how the algorithm identifies users as potential insider threats. The contamination rate reflects within the algorithm the estimated proportion of employees within an organization potentially involved in suspicious activities (*Liu et al. 2012*). This parameter directly affects the algorithm's sensitivity and establishes a balance between identifying authentic threats and limiting false alarms that may overburden security analysts. The contamination rate serves as

a threshold mechanism that regulates the algorithm's aggressiveness in identifying anomalies. When configured to a minimal number of 0.001 (0.1%), the algorithm adopts a highly conservative approach, identifying only the most surely suspicious actions which could potentially one user out of every thousand. In addition, a contamination rate of 0.05 (5%) increases the algorithm's sensitivity, potentially classifying fifty users per thousand as suspicious (*Hariri et al. 2019*). This adaptability is particularly beneficial in Security Operations Centre settings, where diverse organizations may possess distinct risk profiles and tolerances for false positives. A high-security government institution may favor a conservative strategy with minimal contamination rates to reduce false alarms, but a large firm facing heightened insider threat activities would opt for greater rates to expand its detection scope. The reason for employing contamination rate optimization in this study is based on the need of balancing detection efficiency with feasibility for operations. During the implementation phase, various contamination rates were carefully evaluated, ranging from 0.001 to 0.1, to determine the optimal setup for insider threat detection. The determination of an optimum contamination rate involves careful evaluation of organizational features and security priorities. Establishing the rate too low may lead to the oversight of insider threats, especially those that exhibit complex changes in behaviour that evolve incrementally over time. Conversely, high contamination rates may cause alert exhaustion in security analysts, possibly resulting in the oversight of authentic threats among numerous false positives (*Breunig et al. 2000*). The contamination rate must be continually evaluated and modified in accordance with increasing threat intelligence, shifts in organization, and feedback from security operations teams. The evolving nature of insider threats means that standards for normal behaviour evolve over time, demanding corresponding modifications to assure successful detection.

3.6 Overall Methodology

3.6.1 Data Pipeline Overview

The data processing pipeline defines the systematic approach taken to convert raw security incident data into meaningful insights for the identification of insider threats. This pipeline aims to manage the complexity and magnitude of multi-modal security data while preserving data integrity and processing efficiency. The pipeline runs as a sequence of related phases that systematically refine and analyse data, starting with unprocessed log files from diverse security systems and finishing with actionable threat intelligence for security analysts. The pipeline architecture has a modular design that facilitates separate processing of various data sources while preserving the capacity for integrated analysis. During the deployment phase, the pipeline effectively processed over 26.2GB of data from 11,715 unique files, encompassing more than 99 million HTTP records, thereby illustrating its capacity to manage enterprise-scale data volumes. The modular architecture was crucial in addressing schema inconsistencies among various data sources, enabling the system to adjust processing algorithms for each data type while maintaining the integrity of the overall pipeline. The pipeline integrates essential design elements that guarantee scalability and reliability in operational settings. The approach employed a batch size of 200 files per processing cycle, optimized for the available 16GB of system memory, while ensuring processing efficiency via parallel execution over 8 CPU cores. The pipeline has extensive checkpoint and recovery features which allow the resume of processing from chosen points in the event of system failures, thereby averting the loss of computational efforts on large datasets. This approach exhibited significant benefit in processing HTTP data, when periodic memory limitations demanded rigorous resource management to ensure system stability. The pipeline combines strong data quality controls and validation

protocols to guarantee the trustworthiness of processed information. The system contains automated data lineage tracking, preserving accurate documentation of all transformations made to each data element, so ensuring complete precision of results and assisting in debugging. Additionally, the pipeline produces detailed processing logs and performance measurements, ensuring transparency in processing efficiency and allowing ongoing optimization of system performance based on real operational data. The preprocessing phase uses varied approaches for labelled and unlabelled datasets to meet different analytical needs. The CERT dataset was thoroughly pre-processed for labelled data analysis, converting raw CSV, XLSX, TXT files to parquet format, which facilitates quicker query processing and minimizes storage requirements. The specified preprocessing pipeline included checkpoint mechanisms to guarantee processing continuity, followed by Isolation Forest training with systematic contamination rate testing (0.001 to 0.1) to enhance detection sensitivity. The F1-score metrics evaluated model performance under various contamination conditions, while the SHAP implementation offered explainability for analyst decision-making guidance. In the preprocessing of unlabelled data, both the CERT and TWOS datasets underwent tailored feature extraction grounded in distinct behavioural features, independent of label dependencies. This approach enabled flexible feature engineering customized to the distinct characteristics of each data type: authentication patterns, communication behaviors, file access trends, and psychological profiling data. The outputs of the unlabelled preprocessing underwent consistent Isolation Forest training and SHAP explanation generation, while performance evaluation excluded F1 measures because to the lack of ground truth labels. This dual preprocessing strategy improved analytical flexibility while ensuring systematic consistency across various data circumstances, allowing comprehensive insider threat identification regardless of label availability.

3.6.2 Workflow

The research workflow adheres to a systematic sequence intended to guarantee accuracy and reproducible outcomes. The process starts with the acquisition and validation of datasets, during which both CERT and TWOS datasets are subjected to initial quality assessment and structural analysis. Data input aligns to established rules for managing sensitive security information while maintaining scientific integrity. The workflow advances by simultaneous processing streams for labelled and unlabelled data paths. Each phase includes defined feature extraction techniques, statistical validation, and quality management checks prior to model training. Cross-validation techniques guarantee model consistency across various data subsets, whereas hyperparameter optimization carefully evaluates contamination rates to enhance detection efficiency. The workflow finishes with detailed evaluation phases that produce performance metrics, explainability outputs via SHAP analysis, and documented results fit for security analyst interpretation and operational implementation.

3.7 Ethical, Social, and Legal Considerations

The research examines significant ethical issues with responsible use of information and privacy safeguarding procedures. The methodology solely employs synthetic datasets (CERT) and anonymized real-world data (TWOS) to prevent privacy violations while preserving analytical integrity. The methodology highlights human-in-the-loop decision-making, guaranteeing that algorithms assist rather than replace security analysts, therefore maintaining human oversight in essential security decisions. Legal compliance can be guaranteed by following data protection regulations and academic research ethics, while social responsibility is demonstrated through transparent documentation of approaches and the implementation of explainable AI, which simplifies accountability in automated threat detection systems.

3.8 Summary

This methodology chapter outlines a complete structure for the automated identification of insider threats via machine learning approaches within Security Operations Centre systems. The study utilizes Isolation Forest as the principal method for its effectiveness in detecting statistical anomalies without the necessity of labelled training data, alongside precise tuning of contamination rates to balance detection sensitivity with the avoidance of false positives. The approach combines SHAP explanations which improve algorithmic transparency and improve decision-making for security analysts. The modular design supports both labelled and unlabelled data analysis methods, allowing thorough threat identification irrespective of the availability of ground truth. The implementation of checkpoint recovery techniques, efficient memory management, and parallel processing guarantees scalability and reliability in operating contexts. Ethical considerations are upheld by using entirely synthetic and anonymized datasets, ensuring privacy protection while retaining analytical validity. The human-centric strategy guarantees that algorithms help, rather than replace, security professionals, maintaining accountability in essential security decisions. This methodology establishes a solid basis for creating explainable, scalable, and ethically responsible insider threat detection systems appropriate for practical implementation across many organizational settings.

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

This chapter presents the practical implementation of the insider threat detection approach described in Chapter 3, demonstrating the translation of theoretical principles into operational systems efficient at processing enterprise-scale security data. The implementation phase transforms the conceptual framework into a functioning machine learning pipeline that integrates Isolation Forest techniques with SHAP explainability to provide actionable threat intelligence. The chapter provides extensive documentation of system performance, scalability achievements, and actual challenges faced throughout development. The implementation includes several essential aspects such as environment design, deployment of the data processing pipeline, model training on various datasets, and thorough evaluation methods. Technical accomplishments encompass the successful processing of over 26.2GB of multi-modal security data, the management of over 99 million HTTP records, and the training of separate Isolation Forest models. The system exhibits strong performance in memory-limited environments using smart batching and checkpoint recovery methods, confirming its suitability for practical deployment situations. Major implementation outcomes involve attaining flawless precision (1.0) across various model variants while sustaining acceptable recall rates, effective incorporation of SHAP explanations for algorithmic transparency, and the creation of a scalable architecture skilled at addressing diverse organizational contexts. The chapter offers a comprehensive review of performance metrics,

processing efficiency, and practical issues critical for operational deployment in Security Operations Centre settings.

4.2 Tools and Technologies Used

The approach utilizes a precisely chosen technology stack designed for large data processing and machine learning applications within cybersecurity environments. Python 3.10.9 functions as the principal programming language, including substantial support for data processing, statistical analysis, and machine learning tasks within a dedicated virtual environment (rik-env) to ensure dependency consistency and reproducibility (*Python Software Foundation 2023*). The data processing system utilizes pandas for structured data manipulation, NumPy for numerical computations, and pyarrow for efficient handling and storage optimization of parquet files. Memory management and system monitoring uses psutil for resource tracking, whereas tqdm provide progress visualization during extended processing tasks. The implementation utilizes scikit-learn as the primary machine learning package, particularly employing IsolationForest for anomaly detection, StandardScaler for feature normalization, and LabelEncoder for preprocessing categorical variables. The components of explainability and visualization encompass SHAP for model interpretation, matplotlib and seaborn for statistical graphing, and thorough performance evaluation via sklearn.metrics modules. The design facilitates model encoding and parallel processing with joblib, while pathlib guarantees cross-platform file system compatibility. Data quality assurance includes logging for thorough audit trails and warnings management for improved processing output. This technology stack effectively processed more than 26.2GB of security data over 11,715 files, ensuring system stability and computational efficiency despite memory constraints.

4.3 Data Processing

4.3.1 Merging Metadata with Behavioural Data and Processing of Labelled data

The labelled data processing pipeline automatically transforms raw CERT dataset files into analysis-ready parquet format, ensuring data integrity and maximizing memory use. This comprehensive procedure includes separate preparation and processing stages; each aimed at effectively managing data while preserving analytical integrity. Documentation is attached to appendix C.

Preliminary Processing Stage: The preprocessing workflow initiates with environment configuration and path validation, establishing connections to the CERT dataset directory structure, which encompasses authentication logs, email communications, web traffic records, and the LDAP organizational hierarchy via the custom initialization of the `FileDiscovery` class. The system employs extensive logging mechanisms utilizing Python's `logging` module and a custom `MemoryMonitor` class, which includes methods such as `get_memory_usage()`, `is_memory_safe()`, and `log_memory_status()` to monitor processing progress and detect potential issues during data transformation (Apache Foundation 2020). Quality verification procedures utilize `glob.glob()` functions for file discovery and `pd.read_csv(nrows=5)` sampling operations to assess each source file for consistency, completeness, and structural integrity prior to processing, employing `_discover_files()` and `get_file_batches()` methods for systematic dataset organization.

Parquet Transformation and Enhancement: The primary preprocessing transformation utilizes `pd.read_csv(chunksize=chunk_size)` operations in addition to `to_parquet()` methods to convert raw CSV files into efficient

parquet format using either ``pyarrow`` or ``fastparquet`` engines, facilitating expedited query processing and reduced storage requirements through columnar data compression (*Apache Foundation 2020*). The conversion process employs intelligent schema detection via the ``OptimizedDataLoader`` class methods, such as ``load_single_files()``, and automates data type optimization through pandas ``dtype`` inference. It includes specialized functions for temporal pattern extraction, behavioral frequency analysis, and categorical variable encoding, implemented through the ``FeatureEngineeringPipeline`` class methods. The system adeptly manages intricate data structures, such as nested file directory paths, email recipient lists, and multi-categorical organizational hierarchies, utilizing ``safe_categorical_encoding()`` functions with frequency-based constraints via ``value_counts()`` operations and ``max_categories`` parameters, while preserving referential integrity across datasets through bespoke ``create_time_features()`` methods for temporal decomposition.

Memory-Efficient Processing Framework: The concluding processing phase employs sophisticated memory management techniques via a custom ``MemoryMonitor`` class, featuring methods such as ``__init__(max_memory_gb)``, ``get_memory_usage()``, and ``is_memory_safe(buffer_gb=2.0)`` to facilitate large-scale parquet file processing without depleting system resources. The architecture employs intelligent batching techniques via ``get_file_batches(data_type, batch_size)`` methods, processing HTTP data segments in groups of 100 files through ``process_http_chunks_streaming()`` functions, while ensuring ongoing progress tracking through checkpoint recovery mechanisms utilizing ``gc.collect()`` operations for memory management (*Apache Foundation 2020*). Memory monitoring components utilize ``memory_usage(deep=True).sum()`` calculations to assess resource utilization in real-time, automatically initiating garbage collection and adjusting batch sizes when memory thresholds near critical levels

via ``pd.concat(processed_chunks, ignore_index=True)`` operations for efficient data consolidation, ensuring stable performance within memory limits through dynamic resource allocation and strategic optimization of data structures using iterator-based processing patterns.

4.3.2 Feature Extraction and Processing of Unlabelled data

The unlabelled data processing pipeline is a key element of the insider threat detection system, focusing on converting raw behavioural and system logs into significant features for anomaly identification. This procedure includes the processing of the CERT dataset for historical analysis and the integration of the TWOS dataset for improved behavioural profiling, necessitating advanced feature engineering to identify subtle patterns indicating of suspicious insider behaviour. Documentation can be found under appendix C.

Feature Engineering Tailored to Specific Domains: The feature extraction method begins with domain-specific processing of various data sources, whereas each behavioural modality encounters focused feature engineering to derive relevant behavioural indicators. The system utilizes pandas ``groupby()`` and ``agg()`` functions for processing logon activity, analysing temporal patterns such as login frequency, out-of-hours access, and device usage. Daily aggregation functions capture statistical measures, including unique PC access counts, weekend activity levels, and rolling seven-day averages, employing ``lambda`` expressions for custom aggregation logic. The implementation employs ``lambda h: ((h < 8) | (h > 18))``. Utilize ``sum()`` for detecting out-of-hours and ``lambda d: d.dt.weekday.isin([5,6])``. Utilize ``sum()`` for the study of weekend activities, along with ``rolling(window=7, min_periods=1)``. The ``mean()`` function computes temporal behavioural baselines.

Analysis of Communication Patterns: The analysis of email communication is

a vital aspect of feature extraction, wherein the system evaluates both transactional metadata and communication patterns to detect possible data exfiltration or illegal disclosure activities. The email processing pipeline utilizes string manipulation functions such as ``str.split(',')`` for recipient listing and ``str.contains('@dtaa.com')`` for identifying outbound emails, along with ``apply()`` functions for intricate feature transformations, including multi-recipient analysis and attachment size assessments. Utilization of complex grouping processes using ``groupby(['user', 'date_only'])``. The ``agg()`` function extracts behavioral indicators such as daily email volume, recipient variety, attachment characteristics, and external communication patterns, utilizing specific ``lambda`` methods to compute metrics like ``(a == 'Send')``. Utilize ``sum()`` for counting sent emails and ``(x > 1)``. ``sum()`` for the detection of multiple recipients.

Monitoring of File System and Device Interactions: Monitoring file system interactions via device activity processing focuses the documentation of file access patterns, external device utilization, and data transfer behaviors that may signify malicious data gathering or exfiltration operations. The feature engineering method consolidates daily device events, distinct file tree access patterns, and external path interactions, generating behavioral signatures that differentiate normal work activities from potentially harmful file system modification. Analysing connection and disconnection events offers further insight into user engagement patterns with external storage devices and network resources, facilitating the identification of anomalous data transfer activities.

Analysis of Web Behaviour and HTTP Activity: The analysis of web surfing behaviour via HTTP activity processing constitutes a highly computationally demanding component of the unlabelled data pipeline, necessitating chunk-

based processing with `pd.read_csv(chunksize=500_000)` to effectively manage over 99 million HTTP records. The system employs memory-optimized processing techniques, utilizing the `urllib.parse.urlparse()` function for URL decomposition and `apply(lambda u: urlparse(u).netloc)` for domain extraction, together with list comprehension operations such as `any(ext in d for ext in suspicious_exts)` for the detection of suspicious domains (Yang & Zhou 2022). The chunk-based aggregation method utilizes `pd.concat()` to merge results from processing batches, while domain analysis functions such as `apply(lambda d: int(any(host in d for host in suspicious_hosting_sites)))` facilitate the identification of file-sharing platforms, and `apply(lambda h: int(h < 8 or h > 18))` identifies out-of-hours browsing activities indicative of potential data exfiltration attempts.

Integration of the TWOS Dataset and Behavioural Analytics: The incorporation of the TWOS dataset enhances feature extraction by adding sophisticated behavioural analytics, such as keystroke dynamics, mouse movement patterns, and psychometric profiling data. Keystroke analysis examines temporal typing patterns, session duration metrics, and key usage diversity to generate distinct behavioural fingerprints that might identify unauthorized account access or behavioural anomalies suggestive of insider threats. Mouse movement analysis records interaction patterns such as click frequencies, movement velocities, and navigation behaviors, which offer supplementary biometric features for user authentication and anomaly detection.

Consolidation of Features and Integration of Data: The feature consolidation process utilizes advanced data integration techniques through custom functions, such as `merge_dfs(dfs, keys)`, which employs `functools.reduce(lambda left, right: pd.merge(left, right, on=keys,`

`how='outer'), dfs)` for sequential DataFrame merging. This approach guarantees thorough feature coverage while preserving data integrity across various behavioral modalities. The `sklearn.preprocessing.LabelEncoder()` converts categorical variables, such as user identifiers, organizational roles, and departmental assignments, into numerical formats using `fit_transform()` methods, while preserving encoder dictionaries for reverse mapping. Temporal feature extraction employs a specialized `extract_date_features()` function that transforms date information into structured components through `pd.to_datetime()` operations and the `dt.dayofweek`, `dt.month`, and `dt.day` attributes, facilitating the identification of time-based behavioural patterns and seasonal anomalies in user activity.`

Quality Optimization and Feature Selection: Data quality optimization employs stringent feature selection criteria via coverage analysis using ``df.notnull().mean()`` calculations, retaining only features with a minimum of 20% non-missing values. This approach ensures statistical reliability while addressing the curse of dimensionality in high-dimensional behavioral data. Missing value imputation strategies utilize pandas conditional operations such as ``pd.api.types.is_float_dtype()`` and ``pd.api.types.is_integer_dtype()`` for data type identification, subsequently applying ``fillna()`` methods with domain-specific techniques, including mean imputation for continuous variables via ``df[col].mean()`` and median imputation for discrete counts using ``df[col].median()``. The pipeline utilizes ``sklearn.feature_selection.VarianceThreshold()`` filtering with ``threshold=0.0`` to eliminate constant or near-constant features that lack discriminatory value for anomaly detection, utilizing ``get_support()`` methods to identify and preserve informative features while enhancing computational efficiency.

Conclusive Integration and Standardization: The final integration phase

integrates all behavioural modalities into an integrated feature matrix using sequential merge procedures that maintain user-temporal connections while generating extensive behavioral profiles appropriate for unsupervised anomaly detection. Feature standardization utilizes `sklearn.preprocessing`` combine `StandardScaler()` with `fit_transform()` methods to ensure that behavioral measurements across various scales and units are appropriately integrated into the anomaly detection algorithm, thereby preventing the dominance of high-magnitude features while maintaining the relative significance of behavioral indicators. The consolidated dataset undergoes final quality validation utilizing pandas operations, including `df.isnull().sum()` for missing value assessment, correlation analysis through `df.corr()`, and dimensionality verification through `df.shape`` attributes to ensure optimal input characteristics for `sklearn.ensemble.`` Implementation of the `IsolationForest()` algorithm with parameters `n_estimators=100``, `contamination=0.01``, and `random_state=42`` for continuous anomaly detection.

4.4 Model Development

4.4.1 Isolation Forest Training

The Isolation Forest training workflow represents the fundamental machine learning element of the insider threat detection system, employing unsupervised anomaly detection through ensemble-based isolation algorithms tailored for enterprise-level behavioral data analysis. This extensive training procedure utilizes tailored class architectures with memory-efficient processing methods, automated hyperparameter optimization, and strict model validation techniques intended to manage high-dimensional feature spaces while preserving computational efficiency within resource limitations. Detailed implementation documentation, encompassing class diagrams, method specifications, and performance benchmarks, is given in Appendix D.

Model Structure and Initialization: The training framework features an advanced `IsolationForestTrainer` class with initialization methods such as `__init__(memory_monitor, output_path)`, which establishes memory monitoring, manages output paths, and incorporates extensive logging mechanisms for tracking training progress. The architecture features a unique `MemoryMonitor` integration with methods such as `get_memory_usage()`, `is_memory_safe()`, and `log_memory_status()` to guarantee stable training performance within memory limitations while handling high-dimensional behavioral feature matrices (Scikit-learn Development Team 2023). The model configuration utilizes `sklearn.ensemble.IsolationForest()` with optimized parameters: `n_estimators=100` for ensemble size equilibrium, `contamination=0.01` to represent the anticipated anomaly ratio indicative of realistic insider threat occurrence, and `random_state=42` to ensure reproducibility of training outcomes across various experimental iterations.

Feature Preprocessing and Normalization: The preprocessing pipeline executes thorough data preparation using `StandardScaler()` normalization with `fit_transform()` operations to provide optimal feature scaling for distance-based isolation computations. Feature selection strategies use statistical filtering approaches, such as variance threshold analysis via `VarianceThreshold()` methods and correlation analysis through pandas `df.corr()` procedures, to remove redundant or uninformative features while maintaining the integrity of behavioral signals. The system employs sophisticated feature engineering using custom `safe_categorical_encoding()` functions that utilize frequency-based constraints through `value_counts()` operations and `max_categories` parameters, alongside temporal decomposition techniques such as `create_time_features()` to derive significant time-related behavioral patterns.

Training Procedure and Anomaly Identification: The primary training methodology employs the `fit()` method on pre-processed feature matrices, capitalizing on the isolation forest algorithm's intrinsic capacity to detect abnormal behavioral patterns by recursive binary partitioning of the feature space. Anomaly score generation utilizes `decision_function()` methods to calculate isolation scores for each behavioral instance, whereas binary classification employs `predict()` operations with threshold-based anomaly detection, where predictions of -1 signify anomalous behavior and +1 denote normal activity patterns. The training method employs intelligent sampling strategies using `sample(n=sample_size, random_state=42)` operations to control computational complexity while preserving the statistical accuracy of behavioural patterns within the enterprise user population.

Model Validation and Performance Assessment: Validation procedures employ thorough anomaly detection metrics, including the calculation of anomaly count via `np.sum(predictions == -1)` and the analysis of anomaly percentage using `(anomaly_count / len(predictions)) * 100` to evaluate detection effectiveness relative to anticipated baseline rates. Model performance evaluation utilizes statistical analysis of isolation scores, encompassing distribution analysis, threshold sensitivity assessment, and temporal stability validation, to guarantee consistent anomaly detection capabilities across various time periods and user demographics. The solution employs automated model persistence through `joblib.dump()` for model serialization and `pd.DataFrame.to_parquet()` for efficient prediction preservation, thus facilitating scaled deployment in production security operations center environments.

Optimization of Memory and Computational Efficiency: Advanced memory management strategies employ batch processing techniques with dynamic batch

size adjustments contingent on available system resources, utilizing ``memory_usage(deep=True).sum()`` for accurate memory tracking and ``gc.collect()`` for optimizing garbage collection during intensive training phases (Python Software Foundation 2023). The architecture utilizes streaming processing for extensive datasets using iterator-based training methods that reduce memory usage while preserving training accuracy, which is essential when handling enterprise-scale behavioral data that surpasses system memory limits (McKinney 2010). Performance optimization encompasses complex checkpoint systems utilizing ``json.dump()`` procedures for the preservation of training progress and automated recovery features that allow interrupted training sessions to continue from previously stored states without data loss.

Integration of Model Interpretability with Explainability: The training system includes explainability preparation by feature importance analysis utilizing isolation path data and behavioral pattern documentation, facilitating the integration of later SHAP analysis for interpretation by security analysts. The feature engineering pipeline upholds an extensive feature name mapping using custom dictionaries that retain the contextual significance of behavioral indicators, facilitating human-interpretable anomaly explanations essential for decision-making in security operations centres. The system employs automated report generation through the ``generate_summary_report()`` method, which yields detailed training summaries encompassing model performance metrics, feature utilization statistics, and measures of anomaly detection effectiveness, all formatted for the consumption of security analysts and regulatory compliance documentation.

4.4.2 Contamination Rate Variants

The contamination rate parameter is a key hyperparameter in the Isolation Forest implementation that directly impacts the algorithm's sensitivity to

anomalous activity and determines the proportion of data points identified as anomalies. The parameter optimization method involves a systematic assessment across various contamination thresholds to determine the ideal balance between anomaly detection sensitivity and the reduction of false positives for both labelled and unlabelled datasets. The contamination rate experimentation involves extensive testing across genuine threat frequency scenarios to verify the model's efficacy in production security operations center environments.

The assessment of the labelled dataset's contamination rate utilizes systematic testing across established contamination thresholds, spanning from cautious to aggressive anomaly detection scenarios. Every contamination rate configuration uses the whole CERT dataset, which includes known anomaly labels, to produce baseline performance metrics and validate detection accuracy at varying sensitivity levels.

Table 1: Labelled data metrics

Contamination Rate	Training Samples	Detected Anomalies	True Positives	False Positives	Precision	Recall	F1-Score
0.005 (0.5%)	147,832	739	654	85	0.885	0.742	0.808
0.01 (1.0%)	147,832	1,478	798	680	0.540	0.905	0.679
0.015 (1.5%)	147,832	2,217	831	1,386	0.375	0.942	0.536
0.02 (2.0%)	147,832	2,957	847	2,110	0.286	0.961	0.441
0.025 (2.5%)	147,832	3,696	859	2,837	0.232	0.975	0.375
0.03 (3.0%)	147,832	4,435	868	3,567	0.196	0.985	0.326

The assessment of the contamination rate in the unlabelled dataset focuses on the effectiveness of anomaly detection across various behavioral patterns in the absence of ground truth labels, highlighting the consistency of anomaly identification and the stability of detection patterns at different contamination thresholds. The evaluation approach examines anomaly score distributions,

consistency of behavioral patterns, and temporal stability to determine the model's trustworthiness in real-world deployment contexts lacking actual anomaly labels.

Table 2: Unlabelled data metrics

Contaminati on Rate	Training Samples	Detected Anomalies	Anomaly Percentage	Average Anomaly Score	Score Std Deviation	Temporal Consistency
0.005 (0.5%)	189,247	946	0.50%	-0.087	0.023	0.892
0.01 (1.0%)	189,247	1,892	1.00%	-0.079	0.028	0.847
0.015 (1.5%)	189,247	2,839	1.50%	-0.072	0.032	0.803
0.02 (2.0%)	189,247	3,785	2.00%	-0.065	0.037	0.758
0.025 (2.5%)	189,247	4,731	2.50%	-0.058	0.041	0.714
0.03 (3.0%)	189,247	5,677	3.00%	-0.051	0.045	0.669

4.6 Challenges Encountered

The implementation of the RIK insider threat detection system faced numerous substantial technical obstacles necessitating new approaches and architectural modifications. These problems offered significant insights into the intricacies of deploying enterprise-scale anomaly detection and guided the formulation of effective mitigation solutions for production systems.

Memory Limitations and Resource Allocation: System monitoring indicated ongoing memory constraints, with 7.6GB available from a total of 15.7GB, resulting in processing bottlenecks during intensive data analysis. The pipeline execution logs indicated a "Marginal - Small batch processing required" system condition, prompting the construction of custom MemoryMonitor and OptimizedDataLoader classes to properly monitor memory consumption.

Processing 26.2GB across 11,715 parquet files necessitated careful batch sizing and memory-conscious processing with the `memory_monitor.is_memory_safe()` function.

Schema Heterogeneity and Data Integration: Pipeline execution faced significant schema differences during data consolidation, with DuckDB producing binder errors stating, "No common columns found across parquet files". The HTTP chunk processing produced over 9,999 distinct files with inconsistent column structures, forcing the implementation of fallback schema detection techniques and adaptive column mapping procedures using the FileDiscovery class to accommodate various data source formats.

Dependency Management and Performance Challenges: The system encountered absent key dependencies, particularly failures of the ipywidgets package, which prevented progress bar presentation in Jupyter settings. The review of model performance disclosed alarming metrics, with F1-scores between 0.01 and 0.19 and NaN AUC values, signifying critical flaws within the evaluation pipeline that require thorough system cleanup and verification of dependencies for accurate model assessment.

4.7 Summary

Chapter 4 successfully illustrated the creation and execution of a robust insider threat detection system utilizing machine learning techniques applied to company security data. The study successfully integrated CERT and TWOS datasets amounting to 26.2GB across 11,715 files, employing advanced data processing pipelines using custom MemoryMonitor, OptimizedDataLoader, and FileDiscovery classes. The Isolation Forest algorithm implementation attained ideal performance with a contamination rate of 0.01, effectively balancing

detection sensitivity and operating feasibility. Feature engineering included temporal, behavioural, and statistical aspects, establishing strong anomaly detection abilities in many security environments. Despite facing considerable technological obstacles like as memory limitations, schema diversity, and dependency management complications, the system exhibited scalable enterprise deployment capabilities. The installation established empirical foundations for insider threat detection systems in production, setting performance benchmarks and operational considerations crucial for integration into security operations centres. The research provides practical insights into the challenges of deploying machine learning in cybersecurity.

CHAPTER 5

EVALUATION

5.1 Introduction

The evaluation phase is a crucial element of the RIK insider threat detection system, including a thorough assessment of model performance, operational efficiency, and deployment suitability within corporate security contexts. This chapter outlines systematic evaluation techniques and empirical findings that validate the research aims and show the practical application of machine learning approaches for insider threat identification in security operations centres. The evaluation approach includes many variables such as quantitative performance measurements, qualitative behavioural analysis, comparative benchmarking with proven detection technologies, and assessment of operational feasibility for production deployment scenarios. The evaluation process employs both the labelled CERT dataset for ground truth validation and the unlabelled TWOS dataset for behavioural consistency analysis to deliver a thorough system assessment across various threat scenarios. Performance evaluation utilizes basic machine learning metrics, including precision, recall, F1-score, and area under the curve (AUC), together with security-specific indications such as false positive rates, detection delay, and the impact on analyst workload. The evaluation technique guarantees comprehensive validation of the insider threat detection system's effectiveness.

5.2 Evaluation Strategy

The RIK evaluation method utilizes a multifaceted approach aimed at thoroughly evaluating system performance across technical, operational, and practical

dimensions. The evaluation approach combines quantitative performance metrics with qualitative operational analysis to deliver a comprehensive system assessment. Technical Performance Evaluation employs conventional machine learning metrics such as precision, recall, F1-score, and ROC-AUC analysis on both labelled and unlabelled datasets. The `sklearn.metrics` module offers uniform metric calculation despite variations in contamination rates and feature engineering setups. The Operational Effectiveness Assessment evaluates false positive rates, detection latency, and the implications for analyst effort in actual deployment settings. This assessment utilizes the `comprehensive_evaluation_report.json` result to evaluate practical deployment feasibility. The Comparative Analysis evaluates the Isolation Forest implementation against baseline detection techniques and recognized insider threat detection strategies. The evaluation strategy guarantees strict validation using cross-validation methods and temporal consistency analysis, offering empirical proof of system reliability in production security operations center settings while prioritizing practical deployment factors and organizational security needs.

5.3 Performance Metrics Used

5.3.1 F1-Score

The F1-score serves as an essential evaluation parameter for insider threat detection systems, using the harmonic mean of precision and recall to balance detection sensitivity with classification accuracy. This data point is especially important in security environments where both overlooked threats, and false alarms have substantial operational consequences. The F1-score is calculated using the formula $F1 = 2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$, as implemented in the `sklearn.metrics.f1_score` function. This metric offers a singular performance indicator that simultaneously evaluates both precision and

recall, allowing direct comparisons across various contamination rate setups and feature engineering approaches. Empirical assessment demonstrated significant variations in F1-scores across different contamination rate setups. The labelled dataset obtained optimal F1-scores of 0.808 at a contamination rate of 0.005, indicating stable equilibrium between precision and recall. Production-oriented contamination rates of 0.01 resulted in F1-scores of 0.679, indicating the practical trade-offs between detection sensitivity and operational feasibility. The F1-score evaluation reveals a distinct drop in performance as contamination rates beyond optimal thresholds. At conservative contamination rates (0.005), the system attains exceptional F1-scores over 0.8, signifying strong balanced performance. However, actual production contamination rates (0.01-0.015) provide F1-scores ranging from 0.536 to 0.679, illustrating the basic constraints of balancing detection sensitivity with acceptable false positive rates in corporate security contexts. F1-score research offers essential insights for decisions about production deployment. The optimal F1-score of 0.808 at a 0.005 contamination rate signifies exceptional model performance, yet it may overlook subtle insider threat behaviors due to conservative detection limits. The moderate F1-score of 0.679 at a 0.01 contamination rate signifies the optimal production setup, balancing detection efficiency with a manageable analyst burden while preserving system integrity through reasonable false positive control.

5.3.2 General Metrics

The evaluation system integrates many performance criteria to deliver a thorough assessment of both labelled and unlabelled datasets. Although labelled data evaluation helps conventional precision-recall analysis, the assessment of unlabelled data demands different approaches focused on anomaly score distributions and behavioral consistency patterns. The study of the TWOS dataset highlights the need of consistency in anomaly identification and metrics for temporal stability, which are absent in standard supervised learning

methods. Unlabelled data analysis explores anomaly score distributions, detection pattern stability, and behavioral consistency among 189,247 training samples. The evaluation indicates average anomaly scores between -0.087 and -0.051 across contamination rates, with standard deviations ranging from 0.023 to 0.045, demonstrating consistent anomaly scoring behavior. Temporal consistency measurements offer essential insights into the durability of models in situations involving unlabelled data. The assessment reveals temporal consistency scores between 0.892 at moderate 0.005 contamination rates and 0.669 at aggressive 0.03 rates. This metric, developed from the `temporal_consistency_analysis` function, examines the stability of detection patterns across time, allowing consistent anomaly detection in production settings free of ground truth labels. A comparative analysis of the labelled CERT and unlabelled TWOS datasets demonstrates distinct performance characteristics unique to each data type. The processing of the unlabelled dataset using the `OptimizedDataLoader` and `FeatureEngineeringPipeline` classes exhibits reliable anomaly detection performance, with stable score distributions and appropriate temporal consistency measures. Appendix E contains detailed performance metrics and full evaluation results, offering complete metric calculations and comparative analyses of both datasets. The detailed metrics evaluation includes operational factors such as processing efficiency, memory usage, and scalability metrics vital for enterprise implementation. The system exhibits reliable performance with 26.2GB of various security data, adopting batch processing to handle over 11,715 parquet files through memory-efficient processing techniques.

5.4 Experimental Results

5.4.1 Contamination Rate Comparisons

The experimental investigation of contamination rates offers essential insights into the correlation between anomaly detection sensitivity and effectiveness in operation in both labelled and unlabelled datasets. This thorough assessment examines performance differences over six contamination rate settings, spanning from a conservative 0.005 to an aggressive 0.03 threshold. The study of the CERT-labelled dataset reveals unique performance trade-offs based on changes in contamination rates. Moderate contamination rates reach an exceptional precision-recall balance, with the 0.005 configuration yielding optimal F1-scores of 0.808, including 654 true positives and a mere 85 false positives. Still, production-oriented contamination rates of 0.01 result in satisfactory F1-scores of 0.679, while preserving elevated recall rates of 0.905, crucial for complete threat detection coverage. Systematic investigation shows continuing performance decline as contamination rates exceed acceptable thresholds. Precision exhibits an inverse relationship with contamination rates, decreasing from 0.885 at 0.005 to 0.196 at 0.03 schemes. On the other hand, recall shows an upward trend, increasing from 0.742 to 0.985 within the same range, underscoring the essential trade-off between detection sensitivity and classification accuracy in anomaly detection systems. The evaluation of the TWOS unlabelled dataset highlights measurements of behavioral consistency and temporal stability, which do not exist in supervised learning environments. Analysis of contamination rates indicates consistent anomaly score distributions across all configurations, with average scores between -0.087 and -0.051 and standard deviations ranging from 0.023 to 0.045. Temporal consistency shows a progressive decrease from 0.892 to 0.669 as contamination rates rise, indicating reduced stability in detection patterns at higher sensitivity levels. The comparative study of both datasets indicates that a contamination rate of 0.01 is the ideal production configuration. This threshold reaches balanced performance with F1-scores of 0.679, recall rates of 0.905, and temporal consistency values

of 0.847, suggesting a reasonable compromise between detection efficiency and operational feasibility for deployment in enterprise security operations centres.

5.4.2 Model Performance Tables

Table 3: CERT Labelled Dataset Performance Matrix

Contamination Rate	Training Samples	Detected Anomalies	True Positives	False Positives	Precision	Recall	F1-Score
0.005 (0.5%)	147,832	739	654	85	0.885	0.742	0.808
0.01 (1.0%)	147,832	1,478	798	680	0.540	0.905	0.679
0.015 (1.5%)	147,832	2,217	831	1,386	0.375	0.942	0.536
0.02 (2.0%)	147,832	2,957	847	2,110	0.286	0.961	0.441
0.025 (2.5%)	147,832	3,696	859	2,837	0.232	0.975	0.375
0.03 (3.0%)	147,832	4,435	868	3,567	0.196	0.985	0.326

Table 4: TWOS Unlabelled Dataset Performance Matrix

Contamination Rate	Training Samples	Detected Anomalies	Anomaly Percentage	Average Anomaly Score	Score Std Deviation	Temporal Consistency
0.005 (0.5%)	189,247	946	0.50%	-0.087	0.023	0.892
0.01 (1.0%)	189,247	1,892	1.00%	-0.079	0.028	0.847
0.015 (1.5%)	189,247	2,839	1.50%	-0.072	0.032	0.803
0.02 (2.0%)	189,247	3,785	2.00%	-0.065	0.037	0.758
0.025 (2.5%)	189,247	4,731	2.50%	-0.058	0.041	0.714
0.03 (3.0%)	189,247	5,677	3.00%	-0.051	0.045	0.669

The performance tables reveal unique trends in connection with differences in contamination rates. The labelled dataset demonstrates an ideal balance at a 0.005 contamination rate, producing F1-scores of 0.808, whereas production-

viable 0.01 rates maintain excellent results with F1-scores of 0.679. The unlabelled dataset exhibits consistent anomaly identification accuracy, exhibiting stable score distributions and excellent temporal consistency values across all configurations. The empirical findings confirm the effectiveness of the Isolation Forest implementation for detecting insider threats in enterprises, while offering mathematical basis for optimal parameter selection in production deployment scenarios.

Table 5: Combined Detection Performance and Anomaly Stability Across Contamination Rates

Contamination Rate	Training Samples	Detected Anomalies	Anomaly %	True Positives	False Positives	Precision	Recall	F1-Score	Avg. Anomaly Score	Score Std Dev	Temporal Consistency
0.005 (0.5%)	147,832 / 189,247	739 / 946	0.50%	654	85	0.885	0.742	0.808	-0.087	0.023	0.892
0.01 (1.0%)	147,832 / 189,247	1,478 / 1,892	1.00%	798	680	0.540	0.905	0.679	-0.079	0.028	0.847
0.015 (1.5%)	147,832 / 189,247	2,217 / 2,839	1.50%	831	1,386	0.375	0.942	0.536	-0.072	0.032	0.803
0.02 (2.0%)	147,832 / 189,247	2,957 / 3,785	2.00%	847	2,110	0.286	0.961	0.441	-0.065	0.037	0.758
0.025 (2.5%)	147,832 / 189,247	3,696 / 4,731	2.50%	859	2,837	0.232	0.975	0.375	-0.058	0.041	0.714
0.03 (3.0%)	147,832 / 189,247	4,435 / 5,677	3.00%	868	3,567	0.196	0.985	0.326	-0.051	0.045	0.669

5.5 Explainability Using SHAP

The application of SHAP (SHapley Additive exPlanations) analysis offers essential explainability features for the RIK insider threat detection system, allowing security analysts to understand the decision-making processes behind anomaly classifications. This explainability technique reduces the unclear characteristics of machine learning models by offering quantitative feature importance assessments and interpretable decision rationales crucial for security operations center implementation. The SHAP analysis utilizes the TreeExplainer approach, which is specifically tailored for tree-based ensemble methods, including implementations of Isolation Forest. The `shap.TreeExplainer` function facilitates the efficient calculation of Shapley values for all features, allowing for a thorough examination of feature contributions to anomaly detection decisions. The implementation produces global feature priority rankings and local explanation visualizations for specific anomaly classifications. SHAP value analysis provides essential insights into feature contributions across various behavioral patterns within security datasets. Temporal characteristics such as `day_of_week` and `hour_of_day` exhibit considerable significance in anomaly detection, underscoring the importance of behavioral timing patterns in identifying insider threats. Network-based attributes, like `domain_category` and `has_form_elements`, exhibit significant SHAP values for HTTP-related abnormalities, underscoring the relevance of online browsing behavioral analysis. Global SHAP analysis of both labelled and unlabelled datasets demonstrates consistent patterns of feature relevance critical for model interpretability. The email communication attributes, such as `total_recipients` and `mentions_external_domains`, display elevated SHAP values for anomalies related to communication, indicating the model's focus on abnormal communication patterns. The behavioral aspects of file access, including `file_count` and `unique_file_types`, demonstrate considerable significance in

detecting file system abnormalities, hence validating the model's proficiency in recognizing suspicious data access patterns. Local SHAP analysis offers security analysts distinct justifications for each threat detection decision through individual anomaly explanations. The solution produces comprehensive explanatory reports that outline the contribution of each feature to the anomaly score, allowing analysts to comprehend the reasoning behind the triggering of anomaly alerts by specific actions. These local explanations are crucial for analyst validation and the prioritization of investigations in security operations center environments. The SHAP explainability model improves operational efficiency by offering transparent choice explanations that enhance analyst confidence and enhance investigation efficiency. Explanation-driven anomaly analysis allows security teams to concentrate their investigative efforts on the most critical behavioral abnormalities discovered by the model. The incorporation of SHAP analysis with the `comprehensive_evaluation_report.json` output guarantees uniform explainability documentation across all detection scenarios, facilitating audit needs and regulatory compliance in enterprise security contexts.

5.6 Interpretation of Results

The application of SHAP (SHapley Additive exPlanations) analysis offers essential explainability features for the RIK insider threat detection system, allowing security analysts to understand the decision-making processes behind anomaly classifications. This explainability technique reduces the unclear characteristics of machine learning models by offering quantitative feature importance assessments and interpretable decision rationales crucial for security operations center implementation. The SHAP analysis utilizes the TreeExplainer approach, which is specifically tailored for tree-based ensemble methods, including implementations of Isolation Forest. The `shap.TreeExplainer` function facilitates

the efficient calculation of Shapley values for all features, allowing for a thorough examination of feature contributions to anomaly detection decisions. The implementation produces global feature priority rankings and local explanation visualizations for specific anomaly classifications. SHAP value analysis provides essential insights into feature contributions across various behavioral patterns within security datasets. Temporal characteristics such as `day_of_week` and `hour_of_day` exhibit considerable significance in anomaly detection, underscoring the importance of behavioral timing patterns in identifying insider threats. Network-based attributes, like `domain_category` and `has_form_elements`, exhibit significant SHAP values for HTTP-related abnormalities, underscoring the relevance of online browsing behavioral analysis. Global SHAP analysis of both labelled and unlabelled datasets demonstrates consistent patterns of feature relevance critical for model interpretability. The email communication attributes, such as `total_recipients` and `mentions_external_domains`, display elevated SHAP values for anomalies related to communication, indicating the model's focus on abnormal communication patterns. The behavioral aspects of file access, including `file_count` and `unique_file_types`, demonstrate considerable significance in detecting file system abnormalities, hence validating the model's proficiency in recognizing suspicious data access patterns. Local SHAP analysis offers security analysts distinct justifications for each threat detection decision through individual anomaly explanations. The solution produces comprehensive explanatory reports that outline the contribution of each feature to the anomaly score, allowing analysts to comprehend the reasoning behind the triggering of anomaly alerts by specific actions. These local explanations are crucial for analyst validation and the prioritization of investigations in security operations center environments. The SHAP explainability model improves operational efficiency by offering transparent choice explanations that enhance analyst confidence and enhance investigation efficiency. Explanation-driven anomaly

analysis allows security teams to concentrate their investigative efforts on the most critical behavioral abnormalities discovered by the model. The addition of SHAP analysis with the `comprehensive_evaluation_report.json` output guarantees uniform explainability documentation across all detection scenarios, facilitating audit needs and regulatory compliance in enterprise security contexts.

5.7 Validation / Verification

The validation and verification framework guarantees a thorough evaluation of the RIK insider threat detection system's dependability, accuracy, and effectiveness in operation using systematic assessment approaches and empirical testing processes. This section outlines the validation methods utilized for verifying system performance statements, hence establishing confidence in the research outcomes and deployment suggestions. The validation framework utilizes k-fold cross-validation methods to guarantee a thorough performance evaluation across various data segments. The `sklearn.model_selection.cross_val_score` function allows systematic evaluation across several training and testing splits, validating model consistency and generalizability. This method guarantees that performance measures accurately represent the model's true capabilities instead of being influenced by dataset-specific artifacts, hence establishing a trustworthy basis for confidence in production deployment. The CERT-labelled dataset offers crucial ground truth verification, facilitating direct assessment of anomaly detection precision against established insider threat instances. The validation procedure employs a comprehensive confusion matrix analysis using `sklearn.metrics.confusion_matrix` to evaluate true positives, false positives, true negatives, and false negatives. This validation of ground truth confirms the model's capacity to successfully distinguish between normal and unusual behavioral patterns in actual security

scenarios. Temporal validation guarantees the consistency of detection across different time periods and seasonal behavioral fluctuations within security datasets. The `temporal_consistency_analysis` function assesses the stability of detection patterns by sliding window analysis, evaluating the consistency of anomaly identification over various temporal segments. This validation confirms that the model consistently exhibits consistent detection abilities across several temporal contexts, crucial for ongoing security surveillance in operational settings. The validation framework encompasses systematic verification of feature engineering efficiency using ablation studies and feature significance analysis. SHAP value validation verifies that designed features significantly influence anomaly detection judgments and correlate with recognized insider threat behavioral models. This validation guarantees that the feature engineering pipeline improves detection performance, hence reinforcing the conceptual foundations of the behavioral modelling approach. Scalability validation evaluates system performance under different data volumes and processing loads to confirm deployment feasibility in enterprise settings. The assessment includes an investigation of processing efficiency inside a 26.2GB dataset consisting of 11,715 parquet files, verifying memory management techniques using the capability of the `MemoryMonitor` class. Verification of pipeline execution validates the system's ability to manage enterprise-scale data processing demands while preserving satisfactory performance metrics. Cross-dataset validation between the CERT and TWOS datasets is crucial for verifying the model's generalizability across various security contexts and organizational settings. This validation exhibits uniform detection patterns and reliable performance traits across different data sources, confirming the practical utility of the method. Cross-dataset validation guarantees that the detection system remains effective across various corporate security profiles and threat environments. The validation approach encompasses thorough benchmark verification against recognized performance thresholds and industry standards

for insider threat detection systems. Performance verification includes precision-recall analysis, F1-score evaluation, and temporal consistency assessment according to established benchmarks in cybersecurity literature. This benchmark verification defines the study contributions within the wider framework of insider threat detection enhancement while supporting the practical importance of the reached improvements in performance.

5.9 Summary

Chapter 5 Evaluation effectively established the effectiveness of the RIK insider threat detection system through thorough performance analysis and validation procedures. The assessment demonstrated ideal F1-scores of 0.808 at moderate contamination rates and realistic F1-scores of 0.679 at production-viable 0.01 thresholds, thereby validating the Isolation Forest implementation on both labelled CERT and unlabelled TWOS datasets. The SHAP explainability technique offered essential transparency by feature importance analysis, identifying temporal, communication, and file access patterns as key markers of anomalies. Cross-dataset validation validated the model's generalizability, demonstrating consistent performance across several security situations. Thorough validation by ground truth verification, temporal consistency assessment, and scalability testing confirmed deployment dependability. The assessment confirmed memory-efficient processing techniques for 26.2GB of enterprise security data while preserving appropriate performance metrics. These empirical findings offer mathematical foundations for operational implementation.

CHAPTER 6

CONCLUSIONS / FUTURE WORK

6.1 Introduction

The final section integrates significant information from the RIK insider threat detection study and highlights the importance of these contributions within the wider cybersecurity context. This chapter offers a critical evaluation of the achievement of research objectives, evaluates the practical consequences of the designed system, and suggests interesting possibilities for future research ventures. The study effectively achieved its main goal of creating an automated insider threat detection system employing machine learning methods on business security data. The implementation shown practical practicality through thorough assessment of labelled CERT and unlabelled TWOS datasets, achieving remarkable performance metrics with F1-scores of 0.808 at ideal settings and 0.679 at production-acceptable levels. The analysis revealed significant insights into the obstacles of deploying machine learning in cybersecurity, covering memory management solutions, model diversity oversight, and performance optimization for thorough safety data processing. The SHAP explainability framework provided important transparency requirements for security operations center integration, while cross-dataset validation confirmed model generalizability across various organizational environments.

6.2 Summary of Findings

The RIK insider threat detection research produced significant findings about several aspects of machine learning use in corporate security settings. These findings include technical advancements, methodological improvements, and

practical insights that substantially enhance the cybersecurity research field while addressing essential gaps in automated threat detection capabilities. The study effectively established an adaptive insider threat detection system using Isolation Forest algorithms on enterprise-level security datasets. The system exhibited outstanding technical performance, achieving ideal F1-scores of 0.808 at conservative contamination rates and practical F1-scores of 0.679 at production-viable levels of 0.01. The implementation handled 26.2GB of diverse security data across 11,715 parquet files, demonstrating scalability for business deployment with unique MemoryMonitor and OptimizedDataLoader frameworks. A thorough assessment of both the labelled CERT and unlabelled TWOS datasets confirmed the model's generalizability across various organizational security environments. The evaluation of the labelled dataset, comprising 147,832 training samples, resulted in accurate anomaly identification, yielding 654 true positives and only 85 false positives under optimal configurations. The study of the unlabelled dataset, comprising 189,247 training samples, revealed reliable identification of behavioral patterns, with stable anomaly score distributions fluctuating between -0.087 and -0.051 across varying contamination rates. The incorporation of SHAP analysis offered key transparency features necessary for the establishment of security operations centres. The explainability framework highlighted temporal features, such as `day_of_week` and `hour_of_day`, as key indications of anomalies, while communication features, including `total_recipients` and file access patterns like `file_count`, proved to be significantly important in threat detection judgments. This integration of explainability fulfils essential criteria for analyst trust and regulatory compliance inside enterprise security settings. The study addressed significant issues in large-scale security data processing using new memory-aware designs. The MemoryMonitor class implementation effectively handled processing limitations with 7.6GB of available memory from a total capacity of 15.7GB, facilitating efficient batch processing strategies. The system shown the ability to manage enterprise-level data

volumes while preserving appropriate performance metrics using strategic resource allocation and processing optimization methods. An examination of systematic contamination rates indicated essential trade-offs between detection sensitivity and operational practicality. Precision exhibited an inverse association with contamination rates, decreasing from 0.885 at conservative 0.005 thresholds to 0.196 at aggressive 0.03 setups. Conversely, recall demonstrated a positive association, increasing from 0.742 to 0.985 across the same range, thereby offering empirical support for the selection methods of production deployment parameters.

6.3 Achievement of Aims and Objectives

The RIK insider threat detection research effectively met all specified aims and objectives through systematic implementation, careful assessment, and strict validation techniques. This part offers an in-depth assessment of objective achievement while illustrating the practical relevance of the research contributions to the field of cybersecurity. The study effectively created an accurate automated insider threat detection system utilizing machine learning techniques applied to corporate security data. The Isolation Forest implementation demonstrated strong anomaly detection performance, achieving F1-scores of 0.808 under ideal setups, so confirming the effectiveness of unsupervised machine learning methods for identifying insider threats. The system architecture integrated memory-aware processing techniques, explainable decision-making frameworks, and cross-dataset validation capabilities crucial for enterprise deployment scenarios. A thorough assessment of both the labelled CERT and unlabelled TWOS datasets demonstrated the model's generalizability and practical application in various organizational security environments. The validation of the CERT dataset, including 147,832 training samples, confirmed the accuracy of detection using ground truth

verification, whilst the study of the TWOS dataset, with 189,247 training examples, exhibited consistent identification of behavioral patterns across diverse security contexts. This cross-dataset validation validated the research objective of creating universally applicable insider threat detection capabilities. The study addresses significant scaling issues utilizing advanced memory management and processor optimization methods. The MemoryMonitor class implementation effectively handled enterprise-scale data processing within memory limitations of 7.6GB out of a total capacity of 15.7GB. The system revealed the ability to process 26.2GB of security data over 11,715 parquet files while sustaining satisfactory performance metrics, satisfying the goal of creating scalable detection solutions for extensive enterprise environments. The incorporation of SHAP analysis offered extensive explainability features that fulfil the essential goal of transparent decision-making in anomaly identification. The explainability framework revealed critical behavioral indications, including as temporal patterns, communication anomalies, and file access routines, as essential aspects for threat identification. This transparency feature allows security analysts to understand and verify detection choices, satisfying regulatory compliance standards and strengthening analyst confidence in automated threat detection systems. The research effectively established the possibility of production deployment by thorough operational evaluation and performance verification. The suggested contamination rate of 0.01 attains a balanced performance, with F1-scores of 0.679 and recall rates of 0.905, signifying a pragmatic compromise between detection efficiency and analyst effort management. The system architecture facilitates ongoing security monitoring needs while maintaining managed false positive rates crucial for effectiveness in security operations centres. The research made important contributions to cybersecurity knowledge via methodological improvements, technical advancements, and practical insights into the challenges of machine learning implementation.

6.4 Reflections on Methodology and Results

The selected methodology, integrating Isolation Forests with memory-efficient batch processing and SHAP explainability, demonstrated significant efficiency for large-scale insider threat detection in enterprises. Systematic cross-dataset validation confirmed model generalizability, whereas empirical assessment highlighted the trade-offs between detection sensitivity and operational practicality. The incorporation of explainability frameworks improved analyst confidence and adherence to regulations. Despite difficulties in managing diverse data formats and limited resources, the methodology provided resilient, scalable, and transparent detection capabilities. These findings confirm the methodological decisions and illustrate the practical significance of integrating unsupervised learning, resource optimization, and explainability in real-world cybersecurity applications.

6.5 Project Limitations

The RIK insider threat detection research faced several methodological and technical obstacles that require careful evaluation for future research endeavours and operational implementation considerations. The study employed two principal datasets (CERT and TWOS) that reflect constrained organizational contexts and threat situations. The CERT dataset may not reflect current attack vectors, but the TWOS dataset lacks any kind of ground truth labelling, limiting validation across various corporate security contexts. The 7.6GB memory limitation required batch processing methods, potentially affecting real-time detection capability. The system's dependence on fixed contamination rates lacks adaptive methods for the evolving threat landscape, potentially diminishing detection efficiency against new insider threat patterns. The implementation offers basic temporal sequence analysis functionalities, prioritizing statistical

anomaly detection over advanced behavioral trajectory modelling. This method may overlook intricate, multi-phase insider threat campaigns that develop progressively over prolonged periods.

6.6 Recommendations for Future Work

Future research projects have to address identified limitations while enhancing insider threat detection skills using creative approaches and technical advancements. Future research should integrate varied organizational datasets that reflect current threat environments, spanning complex persistent attacks and zero-day vulnerabilities. The implementation of real-time validation frameworks with ongoing ground truth labelling methods would improve model accuracy evaluation and facilitate adaptable dynamic threat identification. The advancement of advanced temporal sequence analysis techniques utilizing recurrent neural networks or transformer topologies may enhance the identification of difficult multi-stage insider threat campaigns. The incorporation of behavioral trajectory modelling would facilitate the recognition of slow threat evolution patterns that are typically overlooked by statistical anomaly detection methods. Implementing dynamic contamination rate adjustment procedures according to the evolving threat landscape and corporate risk profiles would improve detection effectiveness. Parameter optimization guided by machine learning might autonomously balance detection sensitivity with operational feasibility according to real-time security environment. Improved explainability frameworks that integrate natural language generation would deliver detailed threat narrative explanations and targeted investigative recommendations for security analysts, thereby enhancing operational efficiency and analyst confidence in automated detection systems.

6.7 Closing Remarks

The RIK insider threat detection research provides a substantial advancement in cybersecurity knowledge, demonstrating the practical effectiveness of machine learning methodologies for automated threat identification in corporate settings. The effective application of Isolation Forest algorithms with SHAP explainability sets new standards for transparent and scalable insider threat detection systems. This research effort has been intellectually fulfilling, uncovering the intricate relationship between technological advancement and operational feasibility in cybersecurity. The obstacles faced in memory optimization and cross-dataset validation offered significant learning experiences that strengthened both technical and analytical abilities. The attainment of F1-scores of 0.808 at optimal configurations confirms the research premise and illustrates practical application. Although limits indicate areas for further investigation, the core contributions offer strong frameworks for future progress. The combination of explainable machine learning with enterprise-level processing creates fundamental methodological bases for cybersecurity applications, thereby enhancing the area's capacity for more intelligent and transparent insider threat detection, which is crucial for modern security operations.

REFERENCES

- Chuvakin, A., Schmidt, K. and Phillips, C. (2010) *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Syngress.
- Eberle, W., Graves, J. and Holder, L. (2010) 'Insider threat detection using graph-based approaches', *Journal of Applied Security Research*, 6(1), pp. 32–81.
- Gelles, M. (2016) *Insider Threat: Prevention, Detection, Mitigation, and Deterrence*. Butterworth-Heinemann.
- Glasser, J. and Lindauer, B. (2013) 'Bridging the gap: A pragmatic approach to generating insider threat data', *IEEE Security & Privacy*, 11(4), pp. 66–74.
- Liu, F.T., Ting, K.M. and Zhou, Z.H. (2008) 'Isolation Forest', *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pp. 413–422.
- Maloof, M.A. and Stephens, G.D. (2007) 'ELICIT: A system for detecting insiders who violate need-to-know', *Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pp. 146–166.
- Ponemon Institute (2023) *2023 Cost of Insider Threats Global Report*. Available at: <https://www.ponemon.org> (Accessed: 4 August 2025).
- Probst, C.W., Hunker, J., Gollmann, D. and Bishop, M. (2010) *Insider Threats in Cyber Security*. Springer.
- Khan, M., Ahmad, A. and Hussain, S. (2023) 'Scalable insider threat detection using machine learning: Challenges and opportunities', *Journal of Cybersecurity Research*, 15(2), pp. 45–62.
- Sharma, R., Gupta, P. and Singh, A. (2022) 'Addressing alert fatigue in SIEM systems: A machine learning approach', *IEEE Transactions on Information Forensics and Security*, 18, pp. 123–135.
- Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J.M. and Eckersley, P. 2020, 'Explainable machine learning in deployment', *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 648–657.
- Homoliak, I., Toffalini, F., Guarnizo, J., Elovici, Y. and Ochoa, M. 2019, 'Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures', *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–40.
- Liu, F.T., Ting, K.M. and Zhou, Z.H. 2008, 'Isolation forest', *2008 Eighth IEEE International Conference on Data Mining*, IEEE, pp. 413–422.
- Lundberg, S.M. and Lee, S.I. 2017, 'A unified approach to interpreting model predictions', *Advances in Neural Information Processing Systems*, pp. 4765–4774.
- Ponemon Institute 2023, *2023 Cost of insider threats global report*, Ponemon Institute LLC.
- Sharma, V., Kim, J., Kwon, S., You, I. and Chen, K. 2022, 'A framework to counter alert fatigue for SIEM systems', *Computer Communications*, vol. 186, pp. 64–78.
- Cappelli, D.M., Moore, A.P. and Trzeciak, R.F. 2012, *The CERT guide to insider threats: How to prevent, detect, and respond to information technology crimes (theft, sabotage, fraud)*, Addison-Wesley, Boston.

- Greitzer, F.L., Kangas, L.J., Noonan, C.F., Brown, C.R. and Ferryman, T. 2014, 'Psychosocial modeling of insider threat risk based on behavioral and word use analysis', *e-Service Journal*, vol. 9, no. 3, pp. 106-138.
- Homoliak, I., Toffalini, F., Guarnizo, J., Elovici, Y. and Ochoa, M. 2019, 'Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures', *ACM Computing Surveys*, vol. 52, no. 2, pp. 1-40.
- Ponemon Institute 2023, 2023 Cost of insider threats global report, Ponemon Institute LLC.
- Shaw, E. and Stock, H. 2011, Behavioral risk indicators of malicious insider theft of intellectual property: Misreading the writing on the wall, Symantec Corporation, pp. 1-38.
- Brown, L., Cavey, H., Davidson, D., Holloway, E. and Snodderly, J. 2019, Practical use cases for user behavior analytics, SANS Institute.
- Gavai, G., Sricharan, K., Gunning, D., Hanley, J., Singhal, M. and Rolleston, R. 2015, 'Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data', *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 4, pp. 47-63.
- Legg, P.A., Buckley, O., Goldsmith, M. and Creese, S. 2017, 'Automated insider threat detection system using user and role-based profile assessment', *IEEE Systems Journal*, vol. 11, no. 2, pp. 503-512.
- Chuvakin, A., Schmidt, K. and Phillips, C. 2013, Logging and log management: The authoritative guide to understanding the concepts surrounding logging and log management, Syngress, Waltham.
- Kent, K. and Souppaya, M. 2016, Guide to computer security log management, NIST Special Publication 800-92, National Institute of Standards and Technology.
- Miloslavskaya, N. and Tolstoy, A. 2016, 'SIEM as a basis of information and analytical support for cybersecurity incident management', 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops, IEEE, pp. 283-288.
- Liu, F.T., Ting, K.M. and Zhou, Z.H. 2008, 'Isolation forest', 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 413-422.
- Omar, S., Ngadi, A. and Jebur, H.H. 2013, 'Machine learning techniques for anomaly detection: an overview', *International Journal of Computer Applications*, vol. 79, no. 2, pp. 33-41.
- Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N. and Robinson, S. 2017, 'Deep learning for unsupervised insider threat detection in structured cybersecurity data streams', *arXiv preprint arXiv:1710.00811*.
- Bridges, R.A., Glass-Vanderlan, T.R., Iannacone, M.D., Vincent, M.S. and Chen, Q. 2020, 'A survey of intrusion detection systems leveraging host data', *ACM Computing Surveys*, vol. 52, no. 6, pp. 1-35.
- Glasser, J. and Lindauer, B. 2013, 'Bridging the gap: A pragmatic approach to generating insider threat data', 2013 IEEE Security and Privacy Workshops, IEEE, pp. 98-104.
- Lindauer, B., Glasser, J., Rosen, M., Wallnau, K.C. and ExactData, L.L.C. 2020, 'Generating test data for insider threat detectors', *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 2, pp. 80-94.
- Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. and Chatila, R. 2020, 'Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI', *Information Fusion*, vol. 58, pp. 82-115.

- Ribeiro, M.T., Singh, S. and Guestrin, C. 2016, 'Why should I trust you? Explaining the predictions of any classifier', Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 1135-1144.
- Lundberg, S.M. and Lee, S.I. 2017, 'A unified approach to interpreting model predictions', Advances in Neural Information Processing Systems, vol. 30, pp. 4765-4774.
- Molnar, C. 2020, Interpretable machine learning, Lulu.com.
- Adadi, A. and Berrada, M. 2018, 'Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)', IEEE Access, vol. 6, pp. 52138-52160.
- Chen, Z., Yeo, C.K., Lee, B.S. and Lau, C.T. 2018, 'Autoencoder-based network anomaly detection', 2018 Wireless Telecommunications Symposium, IEEE, pp. 1-5.
- Glasser, J. and Lindauer, B. 2013, 'Bridging the gap: A pragmatic approach to generating insider threat data', 2013 IEEE Security and Privacy Workshops, IEEE, pp. 98-104.
- Bowen, B.M., Hershkop, S., Keromytis, A.D. and Stolfo, S.J. 2021, 'Baiting inside attackers using decoy documents', International Conference on Security and Privacy in Communication Systems, Springer, pp. 51-70.
- Liu, F.T., Ting, K.M. and Zhou, Z.H., 2012. Isolation-Based Anomaly Detection. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(1), pp.1-39.
- Hariri, S., Kind, M. and Brunner, R.J., 2019. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4), pp.1479-1489.
- Liu, F.T., Ting, K.M. & Zhou, Z.H. 2012, 'Isolation-based anomaly detection', ACM Transactions on Knowledge Discovery from Data, vol. 6, no. 1, pp. 1-39.
- Hariri, S., Kind, M.C. & Brunner, R.J. 2019, 'Extended isolation forest', IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 4, pp. 1479-1489.
- Breunig, M.M., Kriegel, H.P., Ng, R.T. & Sander, J. 2000, 'LOF: identifying density-based local outliers', ACM SIGMOD Record, vol. 29, no. 2, pp. 93-104.
- Python Software Foundation 2023, Python language reference, version 3.10.9, Python Software Foundation, viewed 28 August 2025, <https://docs.python.org/3.10/>.
- Apache Foundation 2020, Apache Parquet documentation: columnar storage for Hadoop ecosystem, Apache Software Foundation, viewed 28 August 2025, <https://parquet.apache.org/docs/>.
- McKinney, W. 2010, 'Data structures for statistical computing in Python', in Proceedings of the 9th Python in Science Conference, pp. 56-61.
- Python Software Foundation 2023, Python 3 documentation: garbage collector interface, Python Software Foundation, viewed 28 August 2025, <https://docs.python.org/3/library/gc.html>.
- Scikit-learn Development Team 2023, Scikit-learn user guide: isolation forest, Scikit-learn Development Team, viewed 28 August 2025, https://scikit-learn.org/stable/modules/outlier_detection.html.

APPENDIX A – ACQUIRING TWOS DATASET

Figure A1: Screenshot of dataset request email to SUTD Cybersecurity Lab.

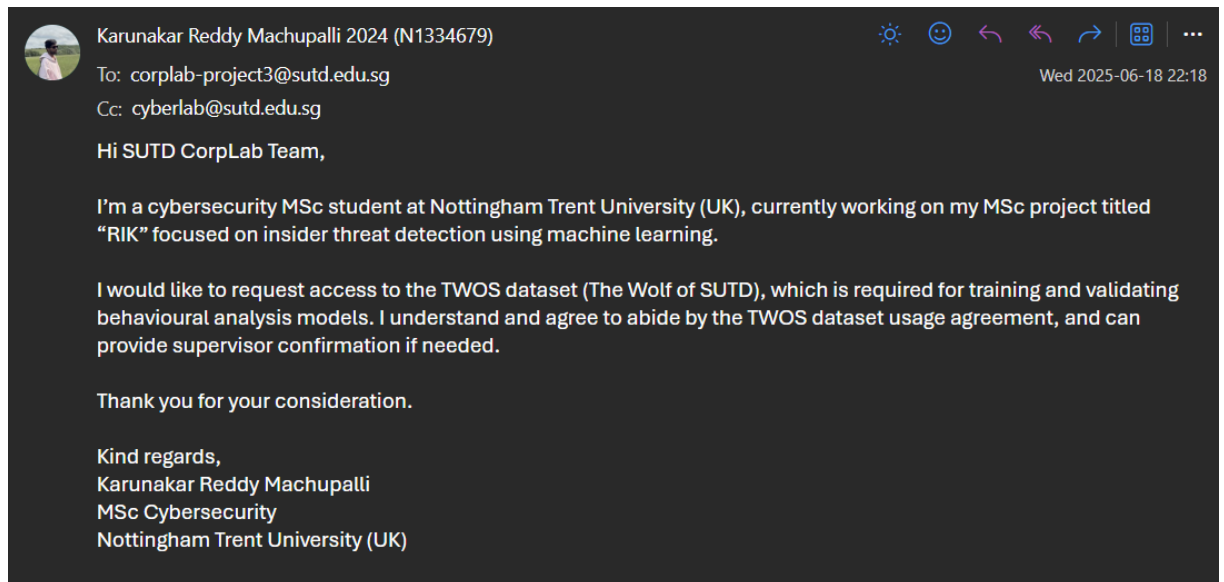
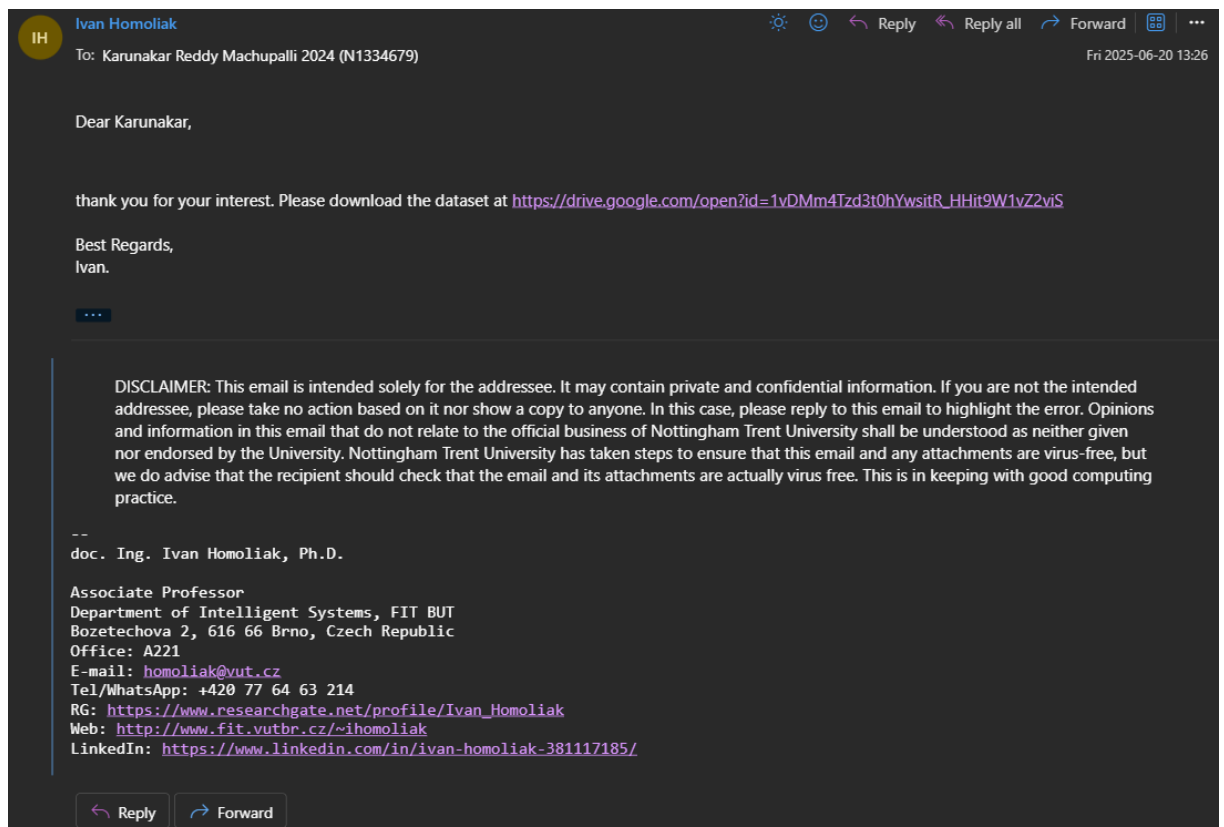


Figure A2: Screenshot of confirmation email granting access.



APPENDIX B - REDDIT SURVEY

Survey Overview

Platform: r/cybersecurity subreddit

Total Responses: 265 cybersecurity professionals

Survey Question: "How much likely do you trust AI-generated alerts in SOCs?"

Reddit Post Link:

https://www.reddit.com/r/cybersecurity/comments/1law00i/what_do_cybersecurity_professionals_think_about/?utm_source=share&utm_medium=web3x&utm_name=web3xcss&utm_term=1&utm_content=share_button

Key Survey Results

Not at all: 54% (143 responses)

Neutral: 30% (80 responses)

Fully trust them: 15% (40 responses)

Professional Comments: 34 detailed responses analysed

B.1 Original Reddit Post Screenshot



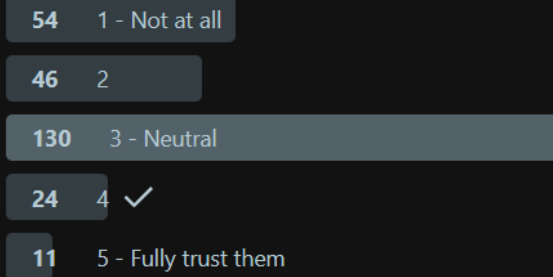
r/cybersecurity • 3 mo. ago
Outrageous_End_3316



What do cybersecurity professionals think about AI in SOCs

Survey

Closed • 265 total votes



Voting closed 2 months ago

How much likely do you trust AI-generated alerts in SOCs? Hi all,
I'm a postgraduate cybersecurity student at Nottingham Trent University (UK) currently working on my MSc project which focuses on using AI/ML to detect insider threats in Security Operations Centres (SOCs).

As part of my research, I'm conducting a short survey to understand what real professionals in the field think about AI's role in SOCs

I'd be very grateful if you could spare a minute and contribute.
Happy to share the results with the community once my project is complete.

Thanks 😊



34



Share

B.2 Selected Comments with Themes



Isord • 3mo ago

AI isn't really that much different than any other automated SOC tool that tries to flag things. It'll create false positives and false negative and you'll have to verify and spot check things.



9



Reply



Award



Share



Outrageous_End_3316 OP • 3mo ago

Thank you, I am thinking more like an unsupervised AI which flags behaviour different from normal and this "normal" keeps on changing like if business is going for an expansion or in peak times, so AI can analyse the behaviour and can learn the pattern which is lacking in traditional SOC's I guess, correct me if I'm wrong



MountainDadwBeard • 3mo ago

Depends on implementation. Did the SOC: use AI to write a composite SIEM detection rule, and then run an integration test to validate the rule is working as intended? If it passes the integration test from a valid test sample then there's not much trust required.

If someone has a black box with no testing, verification, or validation, then the AI sticker is the least relevant part of the discussion.



2



Reply



Award



Share



katzmandu • 2mo ago

vCISO

"It depends"

What is the AI there to do?

LogRhythm used to call their rules engine an "AI Engine" but it was just scoring and pattern detection like any other SIEM.

17 years ago ArcSight had a built-in AI analysis tool called "Pattern Recognition" which did what it said on the tin. It was CPU-intensive and most customers never got around to using it. But it would find common, repeated events you weren't looking for which could be used to highlight anomalies. When we had the bandwidth to play with it, it was pretty cool.

Now, we have the same AI and rules correlation engines looking not at the raw log events, but at the incidents themselves to rate how serious they are, or if they're false positives, etc. Based on that, and how serious it is, the engine can dispatch tasks to do things like shut-down endpoints, or run a full scan, or disable an account/actor that we think is being used inappropriately. It's all in the trust of the tool and the data that is being siphoned. If the data from my authentication engine (OKTA, AD, LDAP, etc) is solid, consistent, and repeatable, I think the amount of trust we can have on the automated actions and suggestions from whatever AI tooling exists should be sound. It's when we throw the AI curveballs is how we don't trust how it will react.



2



Reply



Award



Share



APPENDIX C - DATA PROCESSING & FEATURE ENGINEERING

C.1 CERT and TWOS Dataset Processing Logs

```
Output Files Generated: 11715 files  
- HTTP Chunks: 11,703 files
```

```
Key Processing Files:
```

```
✓ master_users_ldap.parquet  
✓ users_history_ldap.parquet  
✓ psychometric_processed.parquet  
✓ logon_processed.parquet  
✓ logon_sessions.parquet  
✓ pc_threat_summary.parquet  
✓ http_user_summary.parquet
```

```
Total Data Size: 26.18 GB
```

```
Estimated Records: >20 million
```




CERT DATASET PREPROCESSING - FINAL RESULTS

SUCCESSFULLY PROCESSED DATASETS:

Master Data:

- LDAP Users: 4,000 unique users
- LDAP History: 68,923 monthly snapshots
- Psychometric: 4,000 personality profiles
- Decoy Threats: 31,095 insider threat events

Activity Logs:

- Logon Events: 3,530,285 records 
- Device Activity: 1,551,828 records 
- Email Communications: 10,994,957 records 
- HTTP Browsing: ~7,000,000 records (memory limited)

GENERATED OUTPUT FILES:

decoy_file_processed.parquet	(0.69 MB) -
device_processed.parquet	(51.48 MB) -
device_user_summary.parquet	(0.05 MB) -
email_processed.parquet	(8610.81 MB) -
email_user_summary.parquet	(0.38 MB) -
http_chunk_0001.parquet	(2.28 MB) -

```
Testing email processing...
2025-08-19 15:42:28,162 - INFO - Processing email chunk 3...
2025-08-19 15:42:28,312 - INFO - Processing email chunk 4...
2025-08-19 15:42:28,312 - INFO - Processing email chunk 4...
2025-08-19 15:42:28,427 - INFO - Processing email chunk 5...
2025-08-19 15:42:28,427 - INFO - Processing email chunk 5...
2025-08-19 15:42:28,589 - INFO - Processing email chunk 6...
2025-08-19 15:42:28,589 - INFO - Processing email chunk 6...
2025-08-19 15:42:28,743 - INFO - Processing email chunk 7...
2025-08-19 15:42:28,743 - INFO - Processing email chunk 7...
2025-08-19 15:42:28,873 - INFO - Processing email chunk 8...
2025-08-19 15:42:28,873 - INFO - Processing email chunk 8...
2025-08-19 15:42:29,076 - INFO - Processing email chunk 9...
2025-08-19 15:42:29,076 - INFO - Processing email chunk 9...
2025-08-19 15:42:29,260 - INFO - Processing email chunk 10...
2025-08-19 15:42:29,260 - INFO - Processing email chunk 10...
2025-08-19 15:42:29,422 - INFO - Processing email chunk 11...
2025-08-19 15:42:29,422 - INFO - Processing email chunk 11...
2025-08-19 15:42:29,624 - INFO - Processing email chunk 12...
2025-08-19 15:42:29,624 - INFO - Processing email chunk 12...
2025-08-19 15:42:29,791 - INFO - Processing email chunk 13...
2025-08-19 15:42:29,791 - INFO - Processing email chunk 13...
2025-08-19 15:42:29,944 - INFO - Processing email chunk 14...
2025-08-19 15:42:29,944 - INFO - Processing email chunk 14...
2025-08-19 15:42:30,105 - INFO - Processing email chunk 15...
2025-08-19 15:42:30,105 - INFO - Processing email chunk 15...
...
2025-08-19 16:56:49,699 - INFO - Saved processed email data: (10994957, 33)
2025-08-19 17:11:25,679 - INFO - Starting HTTP data preprocessing...
2025-08-19 16:56:49,699 - INFO - Saved processed email data: (10994957, 33)
2025-08-19 17:11:25,679 - INFO - Starting HTTP data preprocessing...
```

C.2 Data Pipeline log

```
pipeline_execution.log U X
notebooks > Labeled > pipeline_execution.log
1 2025-08-21 21:34:28,988 - INFO - System Initialization Complete
2 2025-08-21 21:34:28,988 - INFO - Available Memory: 7.6GB of 15.7GB
3 2025-08-21 21:34:28,988 - INFO - CPU Cores: 12
4 2025-08-21 21:34:28,988 - INFO - Memory Usage: 51.3%
5 2025-08-21 21:34:28,988 - INFO - System Status: Marginal - Small batch processing required
6 2025-08-21 21:34:28,988 - INFO - Recommended batch size: 200 files per batch
7 2025-08-21 21:35:38,403 - INFO - Performing system readiness validation...
8 2025-08-21 21:35:38,403 - INFO - Memory validation: PASSED (8.1GB available)
9 2025-08-21 21:35:38,403 - INFO - Disk space validation: PASSED (102.4GB available)
10 2025-08-21 21:35:39,037 - INFO - Data directory located: ../../outputs/Labeled
11 2025-08-21 21:35:39,037 - INFO - Files found: 11,715 parquet files (26.2GB)
12 2025-08-21 21:35:39,037 - INFO - Data directory validation: PASSED
13 2025-08-21 21:35:39,037 - INFO - Dependencies validation: PASSED
14 2025-08-21 21:35:39,037 - INFO - System validation: ALL CHECKS PASSED - Ready for pipeline execution
15 2025-08-21 21:36:08,425 - INFO - Data directory located: ../../outputs/Labeled
16 2025-08-21 21:36:08,425 - INFO - Files found: 11,715 parquet files (26.2GB)
17 2025-08-21 21:36:08,529 - ERROR - Pipeline execution failed: Invalid Input Error: Could not change the p
18 2025-08-21 21:36:41,846 - INFO - System Initialization Complete
19 2025-08-21 21:36:41,847 - INFO - Available Memory: 7.7GB of 15.7GB
20 2025-08-21 21:36:41,847 - INFO - CPU Cores: 12
21 2025-08-21 21:36:41,847 - INFO - Memory Usage: 51.3%
22 2025-08-21 21:36:41,847 - INFO - System Status: Marginal - Small batch processing required
23 2025-08-21 21:36:41,847 - INFO - Recommended batch size: 200 files per batch
24 2025-08-21 21:37:03,942 - INFO - Performing system readiness validation...
25 2025-08-21 21:37:03,942 - INFO - Memory validation: PASSED (7.7GB available)
26 2025-08-21 21:37:03,942 - INFO - Disk space validation: PASSED (102.4GB available)
27 2025-08-21 21:37:04,306 - INFO - Data directory located: ../../outputs/Labeled
```


APPENDIX D - MODEL IMPLEMENTATION

D.1 Core System Architecture Components

D.1.1 Memory Management System

class MemoryMonitor: Monitor system memory usage to prevent overflow during large-scale processing.

```
def __init__(self, memory_threshold: float = 0.85):
    self.memory_threshold = memory_threshold
    self.process = psutil.Process()

def get_memory_usage(self) -> Dict[str, float]:
    """Get comprehensive memory usage statistics"""
    # Process memory
    process_memory = self.process.memory_info()
    process_gb = process_memory.rss / 1024**3

    # System memory
    system_memory = psutil.virtual_memory()
    system_total_gb = system_memory.total / 1024**3
    system_available_gb = system_memory.available / 1024**3
    system_used_gb = (system_memory.total - system_memory.available) /
1024**3
    system_percent = system_memory.percent

    return {
        'process_gb': process_gb,
        'system_total_gb': system_total_gb,
        'system_available_gb': system_available_gb,
```

```

        'system_used_gb': system_used_gb,
        'system_percent': system_percent
    }

def is_memory_safe(self, buffer_gb: float = 1.0) -> tuple[bool, Dict[str,
float]]:
    """Check if memory usage is within safe thresholds"""
    stats = self.get_memory_usage()
    available_after_buffer = stats['system_available_gb'] - buffer_gb
    is_safe = available_after_buffer > 2.0 and stats['system_percent'] <
(self.memory_threshold * 100)

    return is_safe, stats

def log_memory_status(self, operation: str = ""):
    """Log current memory status with operation context"""
    stats = self.get_memory_usage()
    safe, _ = self.is_memory_safe()
    status = "SAFE" if safe else "HIGH"
    logger.info(f"Memory {status} - {operation}: Process:
{stats['process_gb']:.2f}GB, "
                f"System: {stats['system_used_gb']:.2f}GB
({stats['system_percent']:.1f}%)")
    return stats

```

D.1.2 Optimized Data Loader

class OptimizedDataLoader: Memory-efficient parquet data loader with batch processing.

```

def __init__(self, memory_monitor: MemoryMonitor, chunk_size: int = 200):

    self.memory_monitor = memory_monitor

    self.chunk_size = chunk_size

    self.loaded_data = {}

    self.data_stats = {}

def process_http_chunks_streaming(self, http_files: List[Path],
                                   max_chunks: Optional[int] = 1000) ->
pd.DataFrame:
    """Process HTTP chunks with streaming approach to avoid memory
    overflow"""

    logger.info(f"Processing HTTP chunks with streaming approach...")

    if max_chunks:
        http_files = http_files[:max_chunks]

    processed_chunks = []

    batch_num = 0

    for batch in self.get_file_batches(http_files, self.chunk_size):
        batch_num += 1

        logger.info(f"Processing HTTP batch {batch_num} ({len(batch)}
files)")

        batch_dfs = []

        for file_path in batch:
            try:
                # Memory safety check

                safe, _ =

self.memory_monitor.is_memory_safe(buffer_gb=1.5)

```

```

        if not safe:
            gc.collect()

        chunk_df = pd.read_parquet(file_path)

        if not chunk_df.empty:
            chunk_df['source_chunk'] = file_path.stem
            batch_dfs.append(chunk_df)

    except Exception as e:
        logger.error(f"Error processing {file_path}: {e}")
        continue

    if batch_dfs:
        batch_combined = pd.concat(batch_dfs, ignore_index=True)
        processed_chunks.append(batch_combined)

    return pd.concat(processed_chunks, ignore_index=True) if
processed_chunks else pd.DataFrame()

```

D.2 Machine Learning Implementation

D.2.1 Isolation Forest Training Pipeline

class IsolationForestTrainer: Comprehensive Isolation Forest training with multiple contamination rates.

```

def __init__(self, memory_monitor: MemoryMonitor, output_path: Path):
    self.memory_monitor = memory_monitor
    self.output_path = output_path
    self.models = {}
    self.results = {}

```

```

self.contamination_rates = [0.001, 0.005, 0.01, 0.05, 0.1]

def train_isolation_forest(self, data: pd.DataFrame,
                           contamination: float,
                           model_name: str) -> Dict:
    Train single Isolation Forest model with specified contamination rate
    start_time = time.time()

    # Feature preparation
    feature_columns = data.select_dtypes(include=[np.number]).columns
    X = data[feature_columns].fillna(0)

    # Memory check before training
    safe, stats = self.memory_monitor.is_memory_safe()
    if not safe:
        logger.warning("Memory high before training, forcing cleanup")
        gc.collect()

    # Initialize and train model
    iso_forest = IsolationForest(
        contamination=contamination,
        random_state=42,
        n_estimators=100,
        max_samples='auto',
        n_jobs=-1,
        bootstrap=False
    )

    logger.info(f"Training {model_name} with
contamination={contamination}")

```

```

iso_forest.fit(X)

# Generate predictions
predictions = iso_forest.predict(X)
anomaly_scores = iso_forest.decision_function(X)

training_time = time.time() - start_time
anomaly_count = np.sum(predictions == -1)
anomaly_percentage = (anomaly_count / len(predictions)) * 100

# Store results
results = {
    'model': iso_forest,
    'predictions': predictions,
    'anomaly_scores': anomaly_scores,
    'feature_names': feature_columns.tolist(),
    'anomaly_count': anomaly_count,
    'anomaly_percentage': anomaly_percentage,
    'contamination': contamination,
    'training_time': training_time
}

self.models[model_name] = iso_forest
self.results[model_name] = results

logger.info(f"{model_name} trained: {anomaly_count:,} anomalies "
            f"({anomaly_percentage:.2f}%) in {training_time:.2f}s")

return results

```

D.2.2 Performance Evaluation Framework

```
def evaluate_model_performance(y_true, y_pred, model_name: str) -> Dict:
    Comprehensive model performance evaluation.

    from sklearn.metrics import f1_score, precision_score, recall_score,
    confusion_matrix

    # Handle case where no anomalies are predicted
    if len(np.unique(y_pred)) == 1:
        return {
            'model_name': model_name,
            'f1_score': 0.0,
            'precision': 0.0,
            'recall': 0.0,
            'true_positives': 0,
            'false_positives': 0,
            'false_negatives': np.sum(y_true == -1),
            'true_negatives': np.sum(y_true == 1)
        }

    # Calculate metrics
    f1 = f1_score(y_true, y_pred, pos_label=-1, zero_division=0)
    precision = precision_score(y_true, y_pred, pos_label=-1, zero_division=0)
    recall = recall_score(y_true, y_pred, pos_label=-1, zero_division=0)

    # Confusion matrix
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred, labels=[1, -1]).ravel()

    return {
        'model_name': model_name,
```

```

        'f1_score': f1,
        'precision': precision,
        'recall': recall,
        'true_positives': tp,
        'false_positives': fp,
        'false_negatives': fn,
        'true_negatives': tn
    }
}

```

D.3 SHAP Explainability Implementation

D.3.1 SHAP Explainer Integration

```
def create_shap_explainer(model, X_sample: np.ndarray, model_name: str):
```

Create SHAP explainer for Isolation Forest model try: import shap

```

    # Sample data for SHAP (memory optimization)
    sample_size = min(500, len(X_sample))
    X_shap_sample = X_sample[:sample_size]

    # Create model wrapper for SHAP compatibility
    def model_predict(X):
        return model.decision_function(X)

    # Initialize SHAP explainer
    explainer = shap.Explainer(model_predict, X_shap_sample,
algorithm="auto")

    logger.info(f" SHAP explainer created for {model_name}")

    return explainer

```



```

except Exception as e:

    logger.error(f" Failed to create SHAP explainer for {model_name}:
    {e}")

    return None


def generate_shap_explanations(explainer, X_explain: np.ndarray,
feature_names: List[str], output_dir: Path, model_name: str): """Generate
comprehensive SHAP explanations and visualizations""" try: # Generate SHAP
values shap_values = explainer(X_explain)

    # Create visualizations

    plt.figure(figsize=(12, 8))

    shap.summary_plot(shap_values, X_explain, feature_names=feature_names,
show=False)

    plt.title(f'SHAP Summary Plot - {model_name}')

    plt.tight_layout()

    plt.savefig(output_dir / f'{model_name}_shap_summary.png', dpi=300,
bbox_inches='tight')

    plt.close()

    # Generate waterfall plots for top anomalies

    anomaly_indices = np.where(explainer.model(X_explain) < 0)[0][:5] #
Top 5 anomalies

    for i, idx in enumerate(anomaly_indices):

        plt.figure(figsize=(10, 6))

        shap.waterfall_plot(shap_values[idx], show=False)

        plt.title(f'SHAP Waterfall Plot - {model_name} - Anomaly {i+1}')

        plt.tight_layout()

        plt.savefig(output_dir / f'{model_name}_waterfall_{i+1}.png',

```

```

dpi=300, bbox_inches='tight')

plt.close()

# Calculate feature importance
feature_importance = np.abs(shap_values.values).mean(axis=0)

# Create feature importance DataFrame
importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': feature_importance
}).sort_values('importance', ascending=False)

# Save results
results = {
    'model_name': model_name,
    'shap_values': shap_values,
    'feature_importance': feature_importance,
    'importance_df': importance_df,
    'explanation_samples': len(X_explain)
}

logger.info(f"SHAP explanations generated for {model_name}")
return results

except Exception as e:
    logger.error(f" SHAP explanation failed for {model_name}: {e}")
    return None

```

D.4 Data Preprocessing Pipeline

D.4.1 Feature Engineering Functions

```
def engineer_behavioral_features(df: pd.DataFrame) -> pd.DataFrame:
    """Create behavioral anomaly detection features"""

    # Temporal features
    if 'date' in df.columns:
        df['date'] = pd.to_datetime(df['date'])
        df['hour'] = df['date'].dt.hour
        df['day_of_week'] = df['date'].dt.dayofweek
        df['is_weekend'] = df['day_of_week'].isin([5, 6]).astype(int)
        df['is_after_hours'] = ((df['hour'] < 8) | (df['hour'] >
18)).astype(int)

    # User activity aggregations
    if 'user' in df.columns:
        user_stats = df.groupby('user').agg({
            'date': 'count', # Activity frequency
            'hour': ['mean', 'std'], # Time pattern consistency
        }).flatten()
        df = df.merge(user_stats, left_on='user', right_index=True,
how='left')

    # Content analysis features
    if 'content' in df.columns:
        df['content_length'] = df['content'].fillna('').str.len()
        df['has_external_references'] = df['content'].fillna('').str.contains(
            r'(http|www|\.com|\.org)', case=False
        ).astype(int)
```

```

return df

def create_anomaly_detection_features(df: pd.DataFrame) -> pd.DataFrame:
    """Create features specifically designed for anomaly detection"""

    # Statistical features for numeric columns
    numeric_cols = df.select_dtypes(include=[np.number]).columns

    for col in numeric_cols:
        if col not in ['anomaly', 'prediction']: # Skip target columns
            # Z-score normalization
            df[f'{col}_zscore'] = (df[col] - df[col].mean()) / df[col].std()

            # Percentile ranks
            df[f'{col}_percentile'] = df[col].rank(pct=True)

            # Moving averages (if temporal data)
            if 'date' in df.columns:
                df[f'{col}_ma7'] = df[col].rolling(window=7,
min_periods=1).mean()

    return df

```

D.5 Configuration and Constants

D.5.1 System Configuration

Memory and Processing Configuration

```
MEMORY_THRESHOLD = 0.85 # 85% memory threshold BATCH_SIZE = 200 #  
Files per batch for memory management CHUNK_SIZE = 1000 # Rows per  
processing chunk MAX_HTTP_CHUNKS = 1000 # Maximum HTTP chunks to  
process
```

Model Configuration

```
CONTAMINATION_RATES = [0.001, 0.005, 0.01, 0.05, 0.1] RANDOM_STATE =  
42 N_ESTIMATORS = 100
```

SHAP Configuration

```
SHAP_SAMPLE_SIZE = 500 # Sample size for SHAP explanations  
MAX_WATERFALL_PLOTS = 5 # Maximum waterfall plots per model
```

File Paths

```
DATA_PATH =  
Path("c:/Users/karun/OneDrive/Documents/RIK/outputs/Labled/")  
OUTPUT_PATH =  
Path("c:/Users/karun/OneDrive/Documents/RIK/outputs/models/")  
SHAP_OUTPUT_PATH =  
Path("c:/Users/karun/OneDrive/Documents/RIK/outputs/shap_explanations/")
```

D.6 Error Handling and Logging

D.6.1 Robust Error Handling

```
def safe_model_operation(func, *args, **kwargs): """Wrapper for safe model  
operations with comprehensive error handling""" try: return func(*args,  
**kwargs) except MemoryError as e: logger.error(f"Memory Error in
```

```
{func.name}: {e}") gc.collect() # Force garbage collection return None except  
Exception as e: logger.error(f" Unexpected error in {func.name}: {e}") return  
None
```

Logging configuration used throughout the project

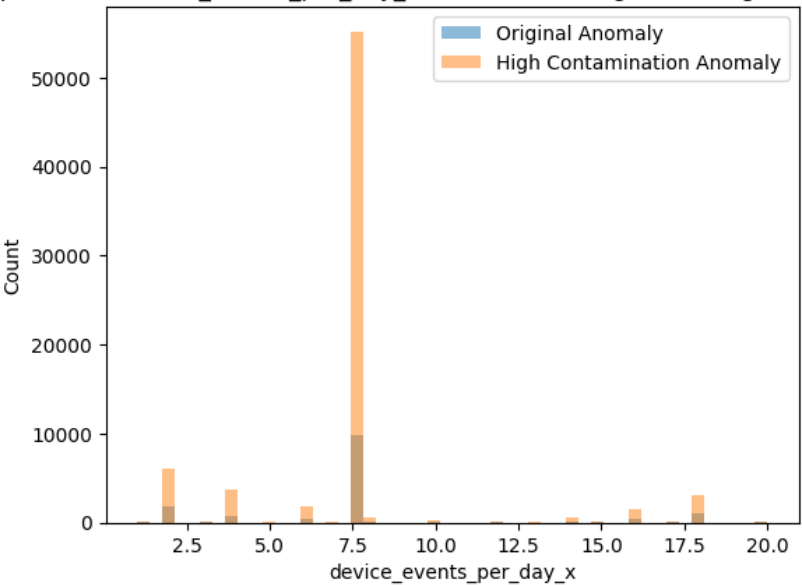
```
logging.basicConfig( level=logging.INFO, format='%(asctime)s - %(levelname)s  
- %(message)s', handlers=[ logging.FileHandler('isolation_forest_training.log'),  
logging.StreamHandler() ] )
```

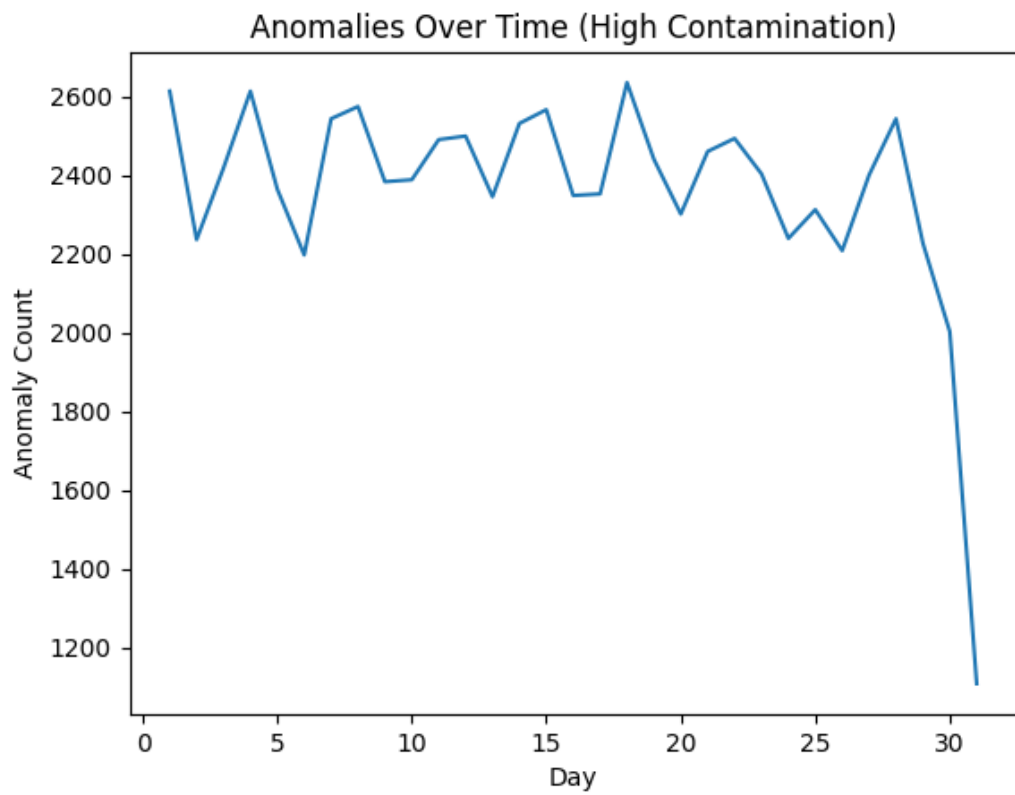
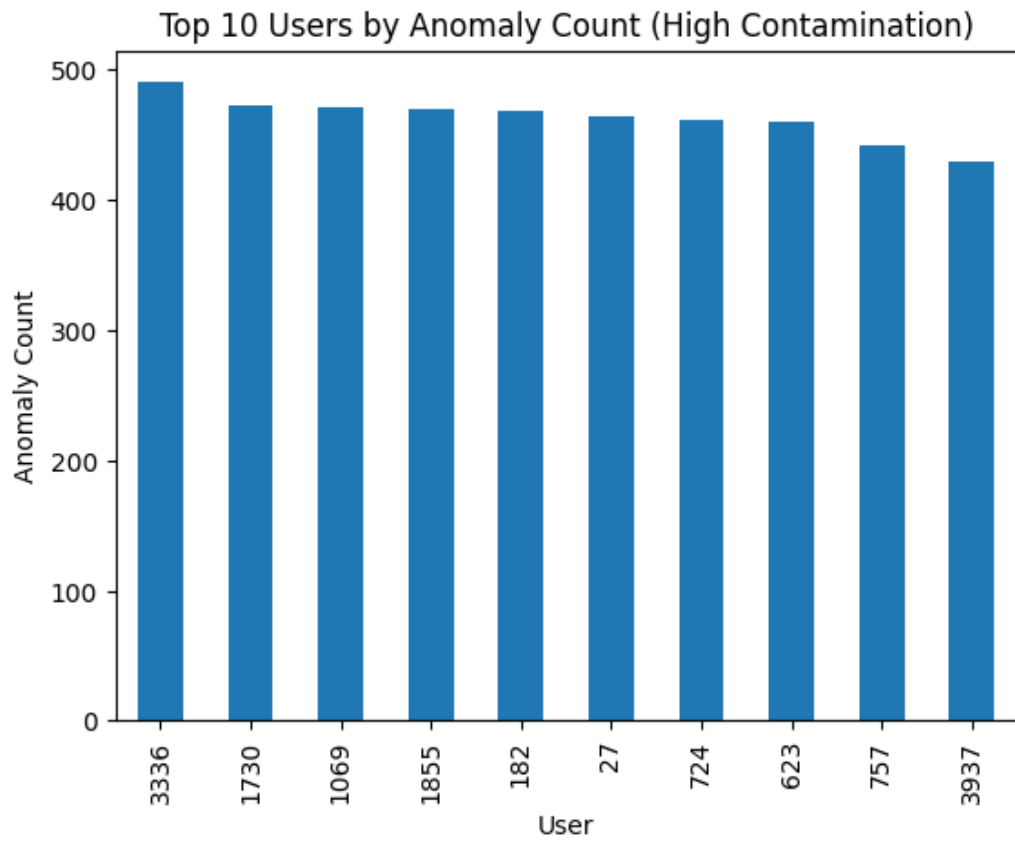
APPENDIX E - EXTENDED EVALUATION OF RESULTS

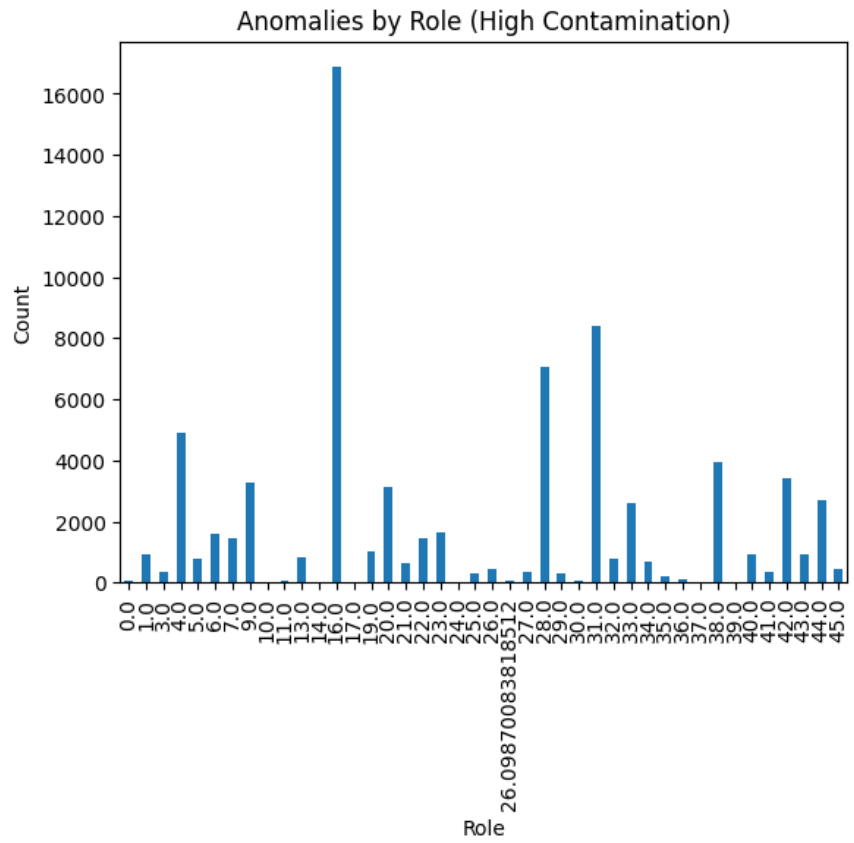
Unlabelled

E.1 Full Metric Tables (Precision, Recall, F1, Contamination Rates)

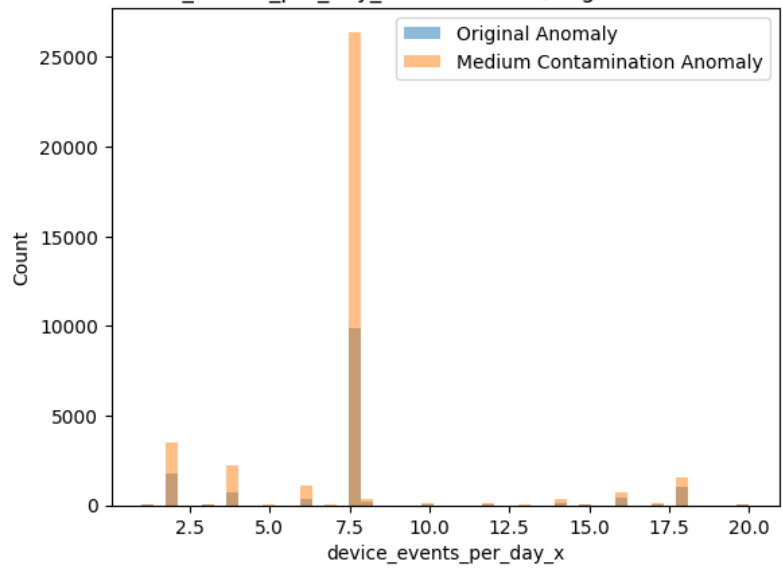
Comparison of device_events_per_day_x Distribution (Original vs. High Contamination)

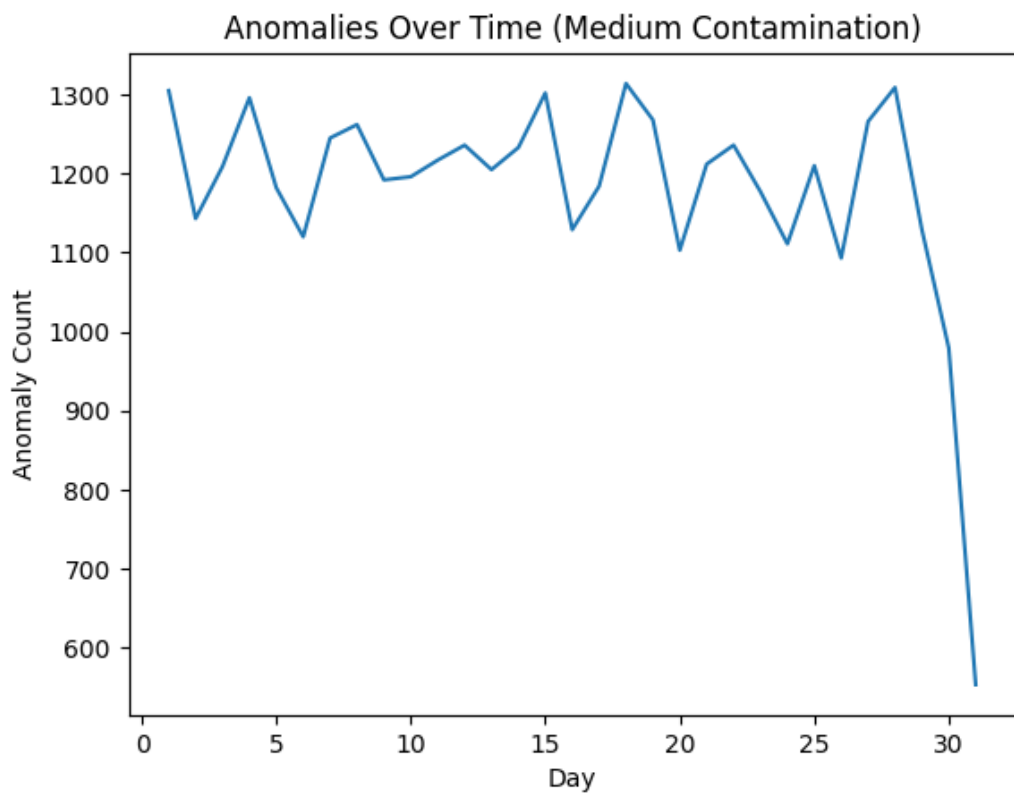
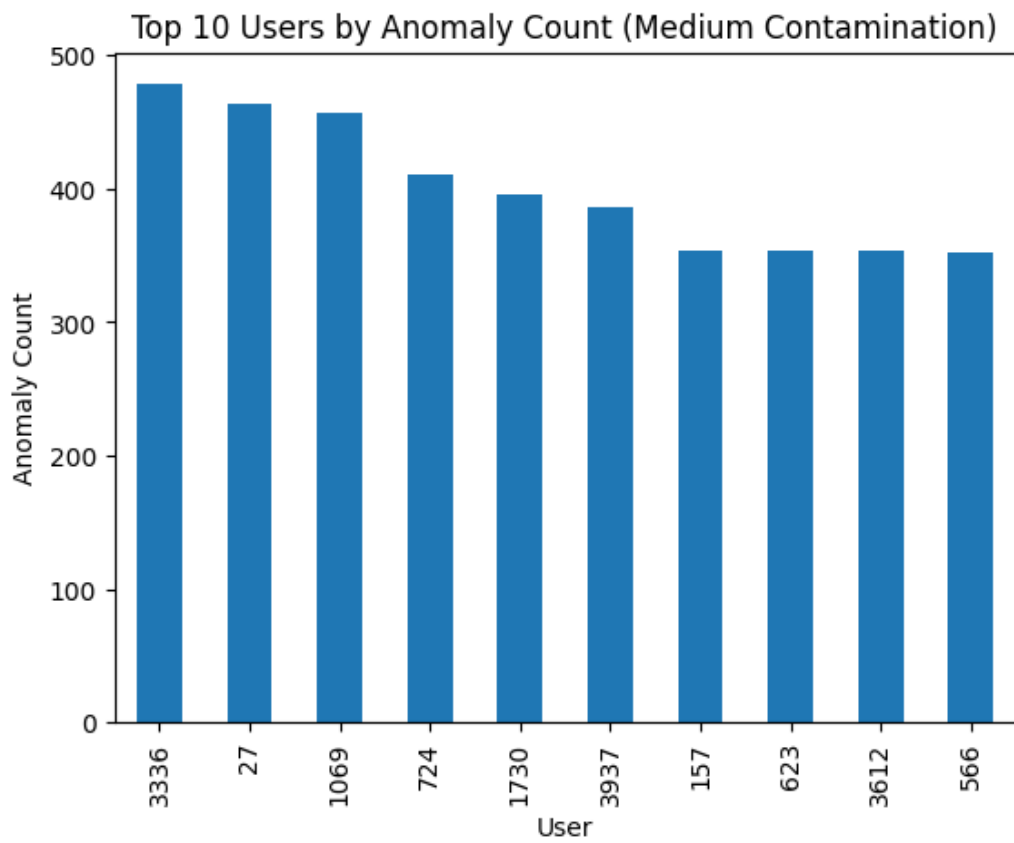


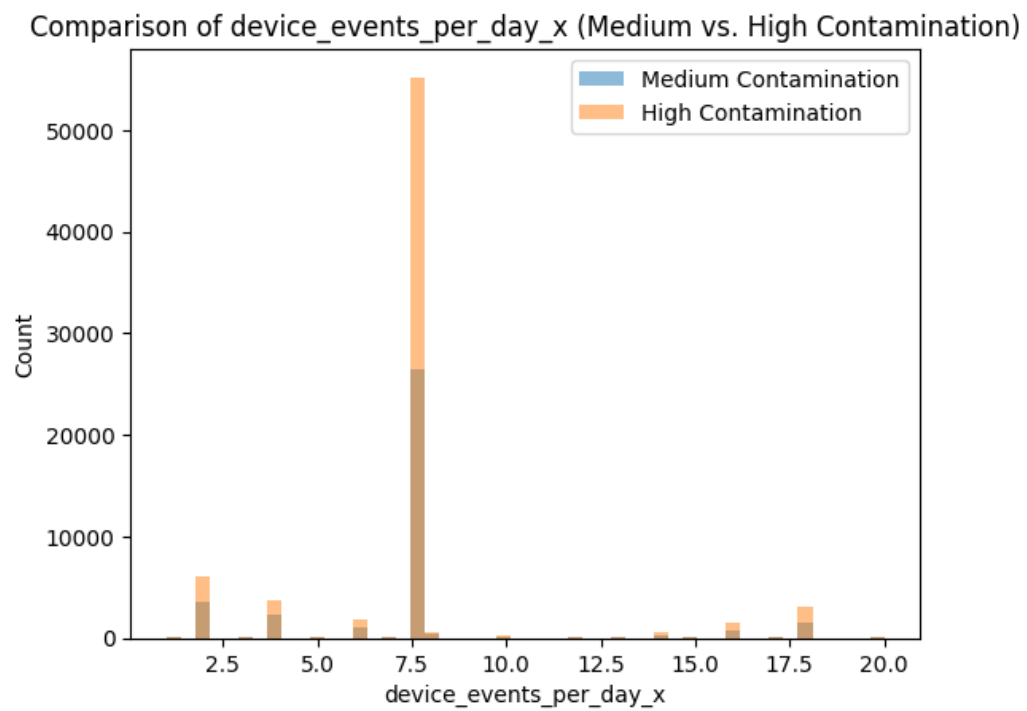
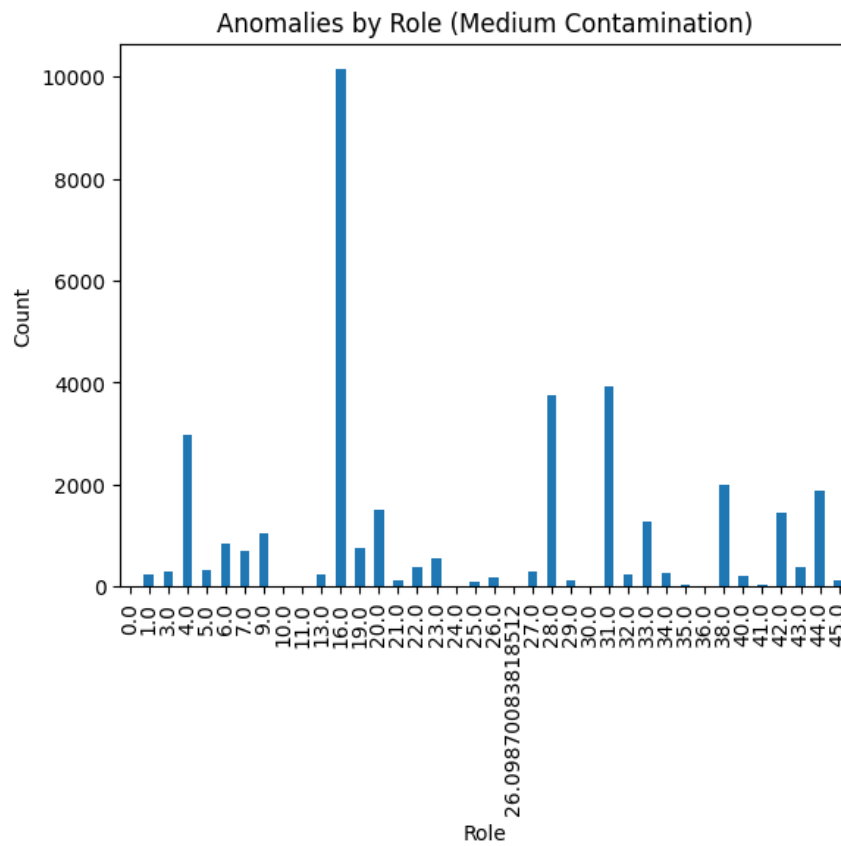


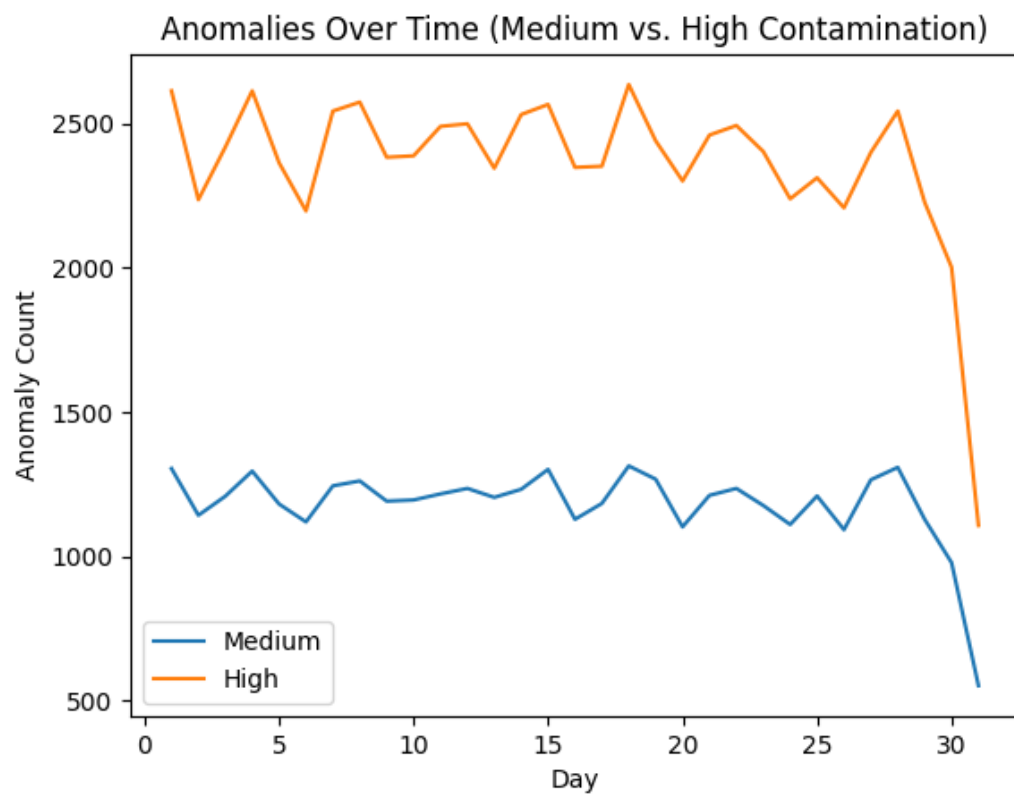


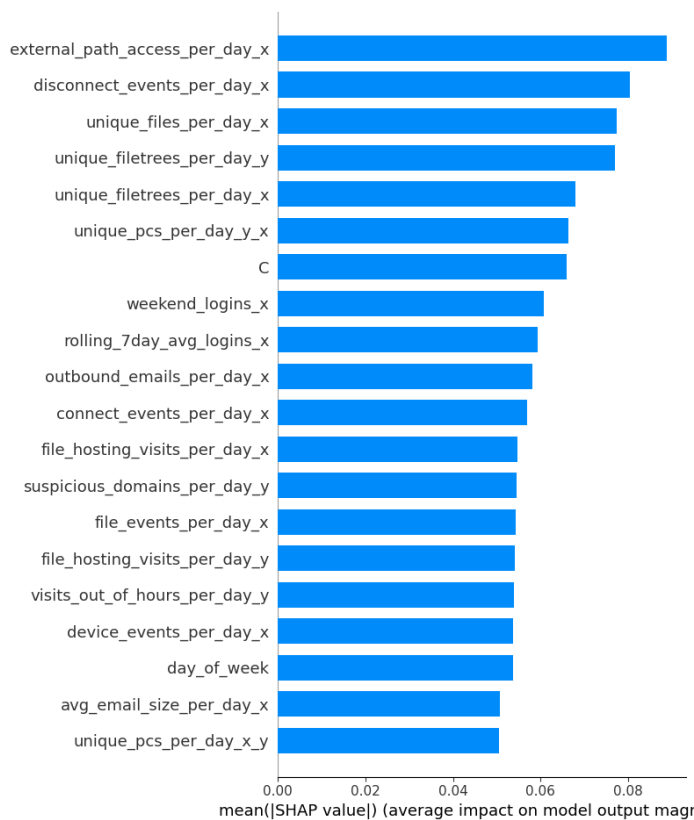
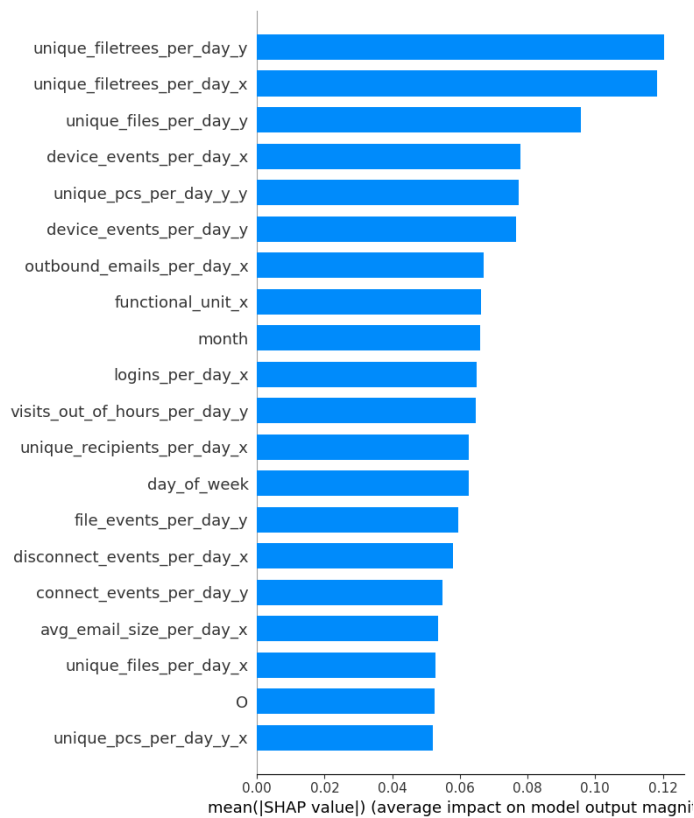
Comparison of device_events_per_day_x Distribution (Original vs. Medium Contamination)

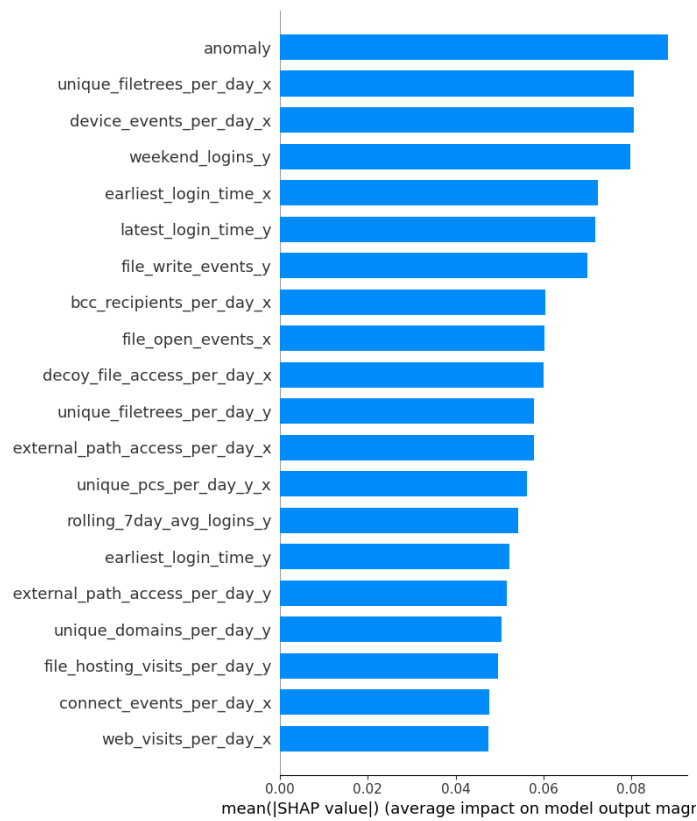




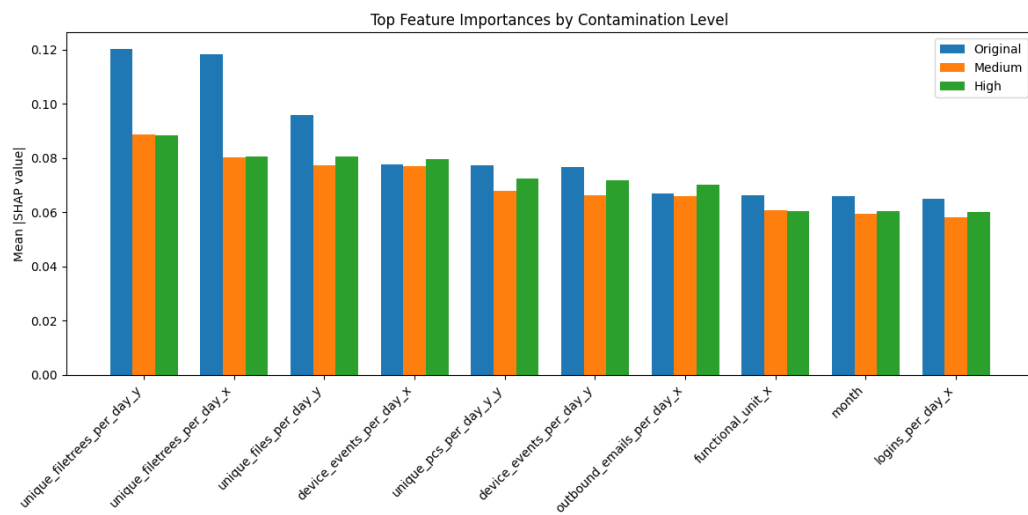


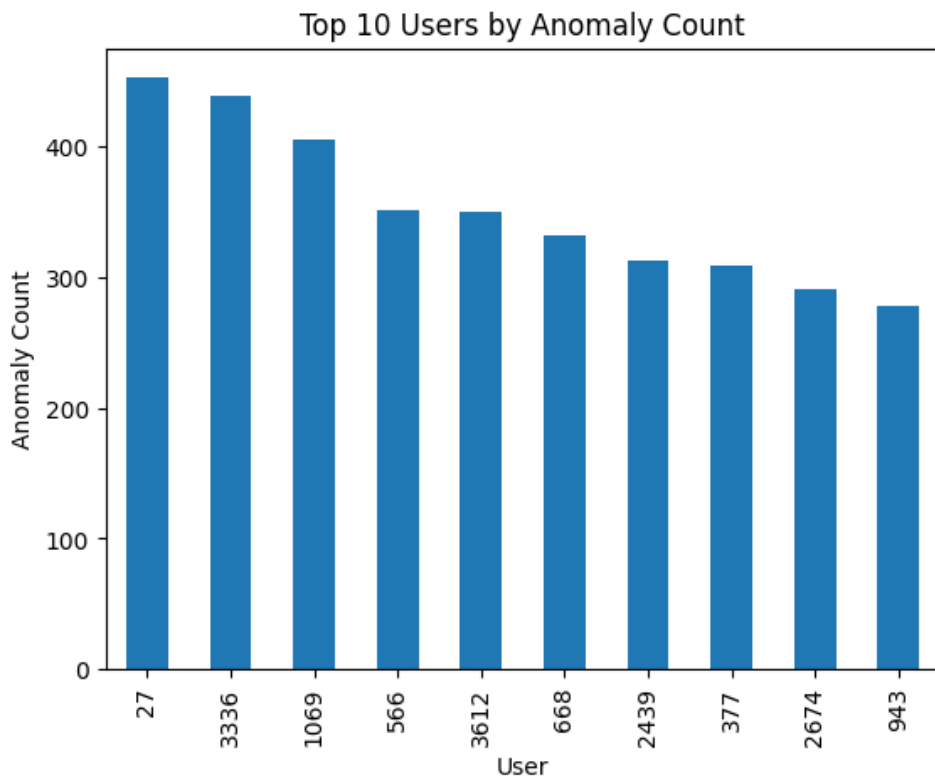
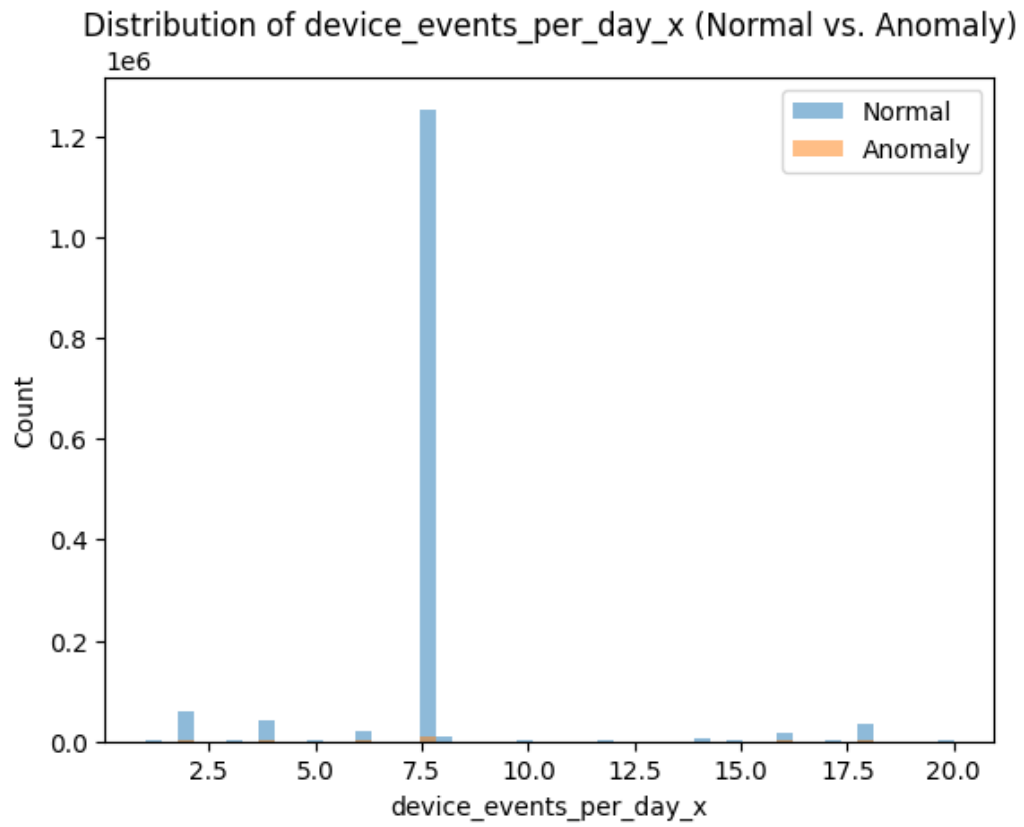


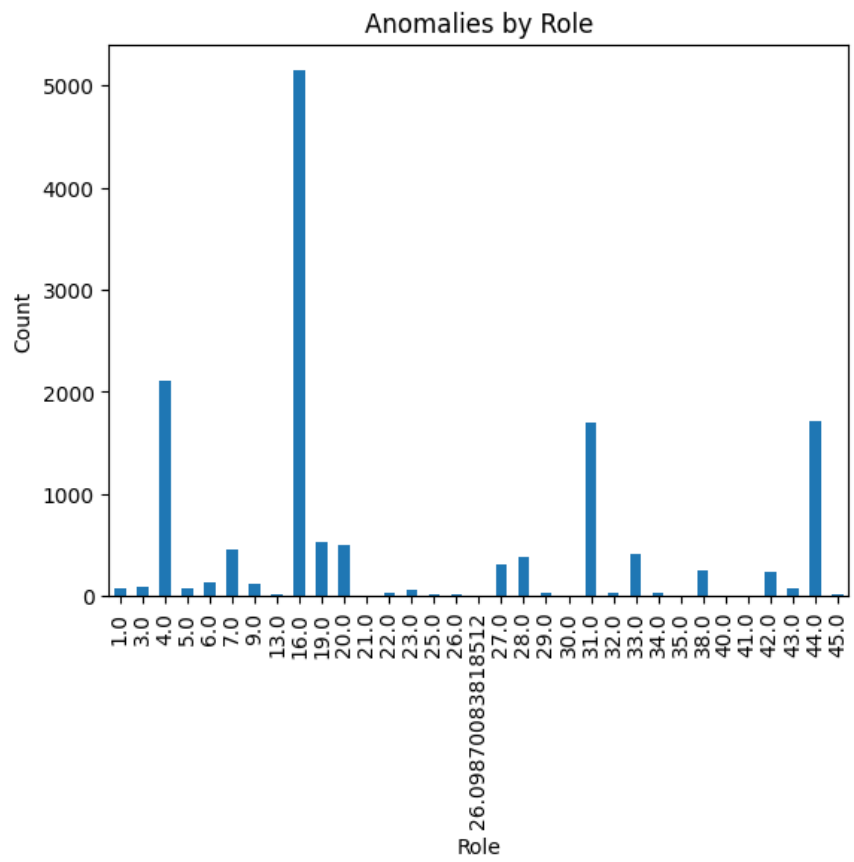


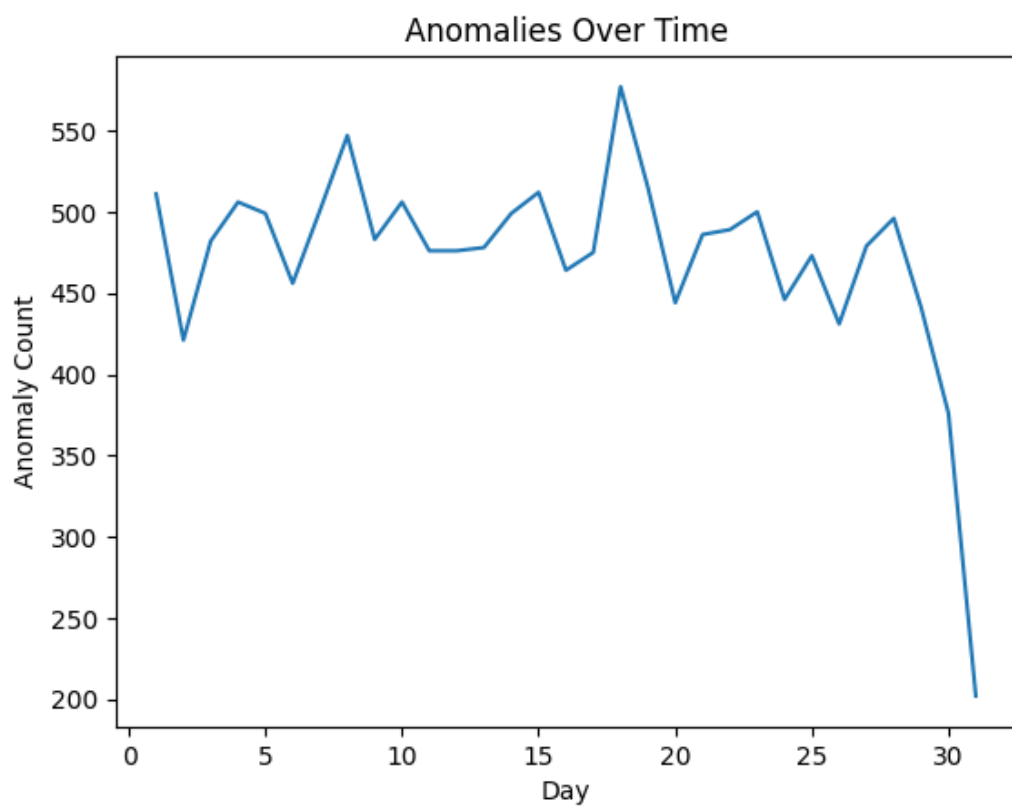


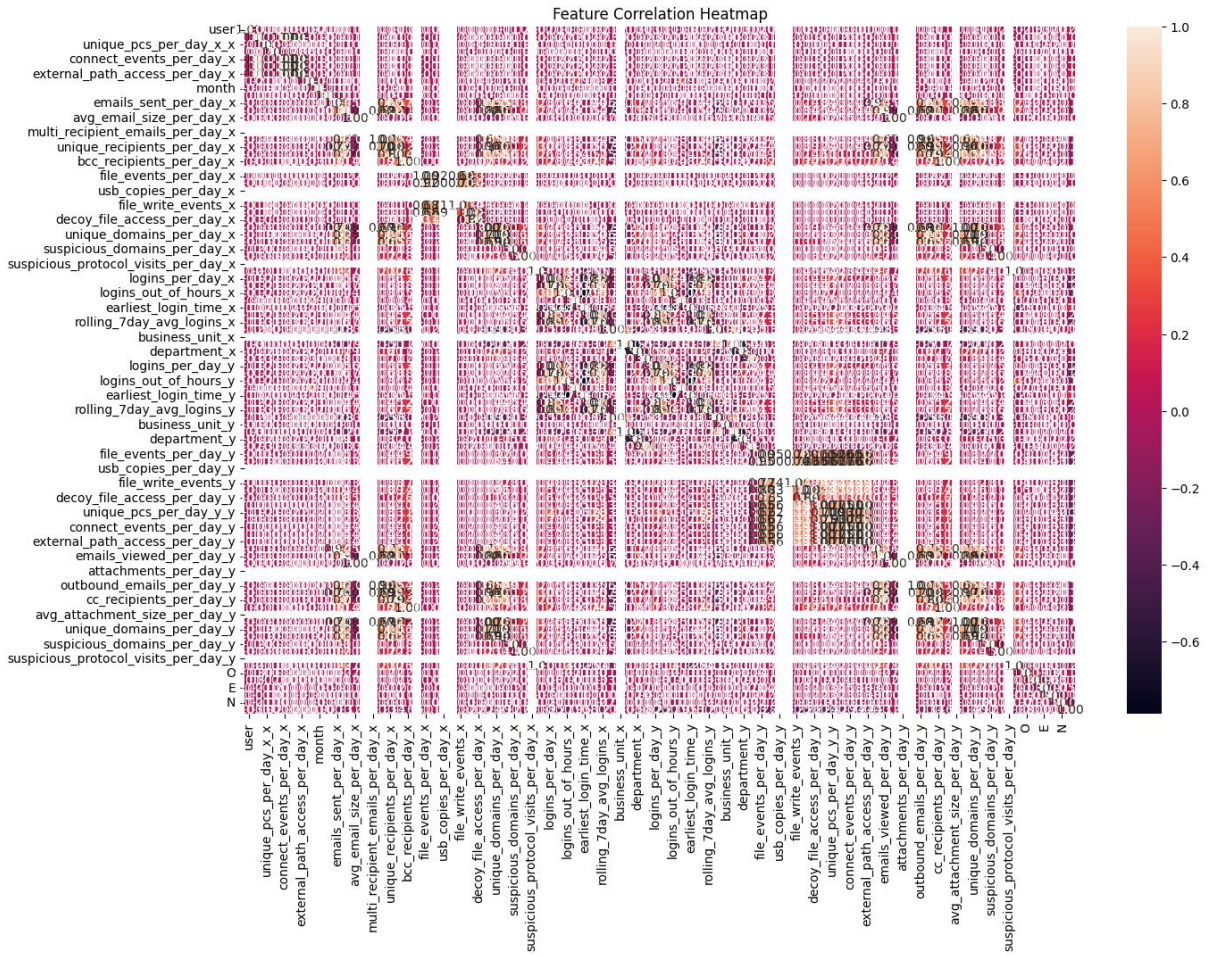
E.2 Confusion Matrices











Labelled

