University of Gloucestershire

School of Computing and Engineering

MSc in Cyber Security

Technical Computing

Dissertation 2023/2024

Deep Learning and Big Data Analytics for Anomaly Detection in The Insider Threat Platforms.

Name: Bhavik Patel

Enrollment No: S4326657

Supervisor: Dr Abu Alam

August 2024

Declaration

This dissertation does not violate the ethical guidelines outlined in the university's handbook for research ethics since, with the exception of instances when it is indicated otherwise by reference or recognition, it is the result of my own effort. I agree that the university may, at its discretion, make it accessible for reference in any and all media by any and all currently known or developed means in the future, provided that any quotations or references to this work made by any individual are properly attributed.

Abstract

In the introduction and literature review chapters, the high and emerging risks of insider threats have been highlighted. A focus on the use of Big Data Analytics and Deep Learning along with the overall emphasis on detection and reduction of false positives is made. In the reviewed literature, the focus is made on the following: the conventional approaches used, Big Data Analytics application, as well as the advantages and limitations of integrating both of these technologies. Current research limitations are highlighted, stressing the importance of more flexible, in-real-time approaches and concerted efforts toward creating a uniform system of measuring the effectiveness of insider threat systems. The method chapter explains the process for designing an automated analytic tool to identify insider threats. Thus, adopting an exploratory research approach and using secondary data sourced from Kaggle, it has employed Convolutional Neural Networks (CNNs) and classifiers. Evaluation measures pertain to the outcomes of the performance, whereas ethical issues relate to the appropriate use of AI systems and data protection. Insider threat analysis is the main subject of this research, and particularly, it deals with the application of deep learning with big data analytics for insider threat detection based on anomaly detection. It compares it with several machine learning categorizations namely Random Forest, KNN, Gaussian NB, MLP, Decision Tree, and CNN. Consequently, the paper showcases how these models can work in conjunction to detect the insider threat while discussing its findings via data analysis and visualizations, which form the basis for the models' future improvements and real-world implementation.

Keywords : Insider threat , Deep learning , Big data , Cyber security , Anomaly detection

Acknowledgements

At the opening of my thesis, I would like to express my gratitude to the members of the British Aerospace Systems (BAE) Insider Threat Team for their help and advice during this study, and to Dr. Abu Alam for his encouragement and support as my supervisor. To my mentor Sumit Thakkar, whose support and sense of humour have been an invaluable source of solace during my stay at the institution. By doing this, I aim to inspire him to push himself to become a better version of myself. Not to be forgotten is my family, without whom I could not have accomplished what I had believed was impossible just two years ago. I am eternally grateful for their love, support, and affection. Thank you.

Contents

List of Figures

Chapter 1

Introduction

Insider security threats therefore have been identified as a key risk to organizations in the digital age. External threats are from people who still have access to the organization's information and its assets, thus making them capable of posing threats intentionally or through negligence. Standard approaches to securities and protection that include rules, and statistical anomaly detection were found to be inadequate in identifying insider threats and the many related behaviors (Sharma et al. 2020). These techniques are generally preventative and are unable to adapt fast enough to many of the strategies used by insiders intent on doing harm. Big Data Analytics and Deep Learning are new promising directions in improving the efficiency of insider threat detection. Big Data Analytics helps in dealing with huge volumes of data that can flow in real-time and reveal possible anomalies that can suggest fraud.

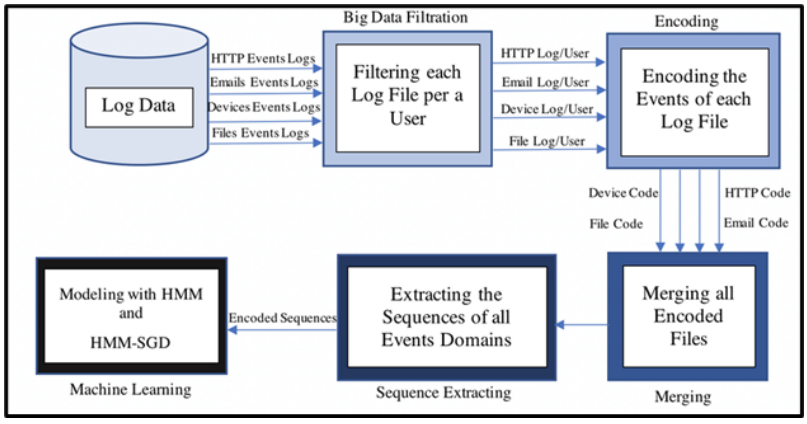


Figure 1.1: : General platform of the insider threat detection system (Saaudi et al. 2019)

Deep Learning describes the nature of the relationships and patterns in the data and helps to learn from large data sets by finding hidden patterns that are difficult to detect by traditional techniques for Big Data Analytics, thus shifting from detection to proactive threat prevention

(Nassar & Kamal 2021). Such systems can also be trained to learn from fresh information, be better prepared in existing threat paradigms, and mitigate false alarms. This research seeks to utilize these advanced technologies in developing an efficient and effective implementation of insider threat countermeasures to enhance security within organizations.

1.1 Problem Statement

How can an organization identify and lessen insider threats using an automated analytic tool?

1.2 Research Questions

This research proposal will aim to answer the following:

1. How can automated analytic tools shield organizations against insider threats?
2. How might the application of deep learning techniques reduce the number of false positives in insider threat detection?
3. How effectively can the developed platform identify anomalies in real-time and adapt to new insider threat strategies?

1.3 Objectives

Aim
The main goal of the study is to improve the detection of insider threat and mitigation approaches by creating an automated analytic tool based on the Deep learning methods and existing big data analytics.

Objectives

1. Conduct a literature review to identify the current approaches to using automated analytic tools for insider threat mitigation.
2. Implement deep learning techniques to reduce false positives in insider threat detection.
3. Evaluate the platform's ability to identify anomalies in real-time and adapt to evolving insider threat strategies.

1.4 Scope of the Research

This research encompasses the exploration of various analysis methodologies and approaches used in detecting insider threats, with the goal of mitigating or preventing such threats. Additionally, it involves the development and evaluation of an automated analysis tool utilizing a comprehensive dataset sourced from BAE Systems over a three-month period, including email, removable media, and print data.

1.5 Ethical Research

All data provided by BAE Systems complies with applicable laws and regulations, including data protection and GDPR. The research adheres to the university's guidelines, ensuring original work and proper citation. As the research focuses on anomaly detection and deep learning methodologies, direct user interaction is not involved.

1.6 Practical Research

In addition to developing a tool for autonomous threat analysis, this research aims to provide users and organizations with a systematic analysis of anomaly detection techniques and machine learning algorithms. This insight will enable users to enhance their monitoring and analysis capabilities within the Insider Threat ecosystem.

1.7 Conclusion

Malicious insiders employ various strategies to extract confidential data from organizations, necessitating effective detection mechanisms. This research seeks to develop a user-friendly solution for employee anomaly detection to identify insiders before data is compromised.

1.8 Structure of the Dissertation

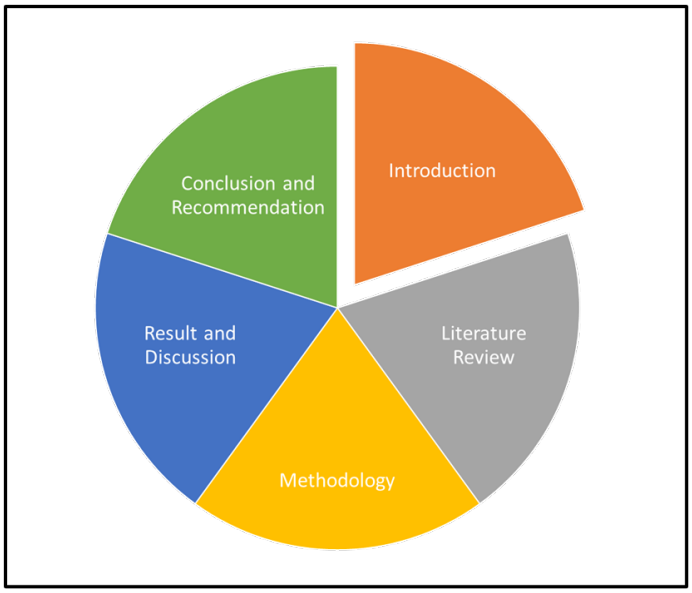


Figure 1.2: Structure of the Dissertation (Self Created)

Chapter 2

Literature Review

2.1 Introduction

This literature review aims to discuss the details of the insider threat detection process and highlight how it has developed, with the main focus on traditional approaches and their drawbacks. It looks into the application of Big Data Analytics and Deep Learning in advancing cybersecurity operations. In this regard, the review focuses on the integration of such innovative technologies pointing out their effectiveness in enhancing the real-time assessment of abnormalities while decreasing the number of false alarms. Further, it reveals the present-day issues and indicates future modifications to fine-tune the effectiveness of automatic analytical solutions to counter insider threats.

2.2 An Overview of Traditional Insider Threat Detection Methods

Most of the existing approaches to tackle insider threats are based on simple rule systems and statistical approaches for anomaly detection. Rule-based systems work based on a set of rules that aim at alerting activities that are out of order. As per the views of (Ning et al. 2021), these systems are easy to apply and analyze and for those reasons, they appealed at the beginning to security operations. Statistical Anomaly detection, on the other hand, focuses on the violation of statistical standard results like users' access patterns, or data transfer rates. However, these methods have various drawbacks when utilized in the detection of complicated and nuanced insider threat behaviors. Malicious insiders are people who have legitimate access to the systems, and they are accustomed to the organizational security frameworks; thus, they can easily bypass rule-based systems, which are rigid and unable to change as the threats do. Statistical anomaly detection, despite being able to detect discrepancies as such, has a problem with a high number of false positives. These false alarms not only consume security resources but on the other hand, cause inefficiency in threat response as they occupy the security team with many unnecessary alarms.

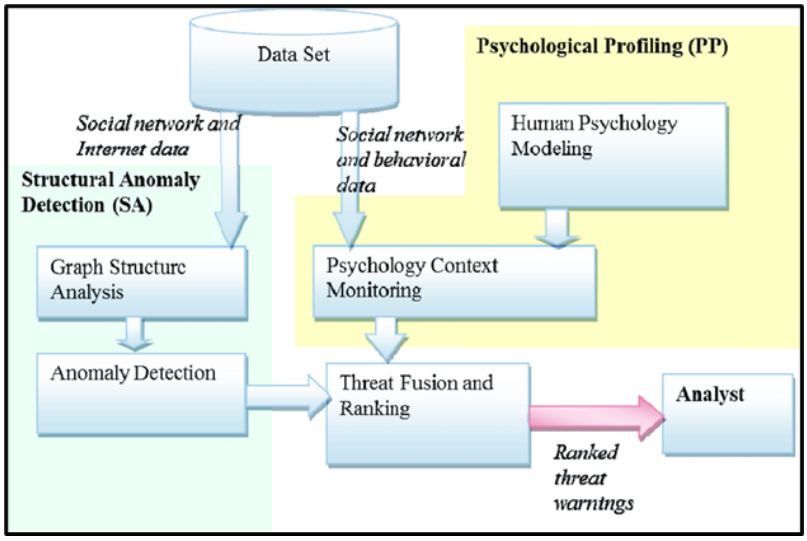


Figure 2.1: Architecture for detecting insider threats using Structural Anomaly (Brdiczka et al. 2012*a*)

Furthermore, it is noteworthy to establish that proactive approaches are optimal in general due to the fact that traditional strategies are reactive and only notice threats after they have occurred or when activity levels are high (Hariharan et al. 2020). This reactive approach obstructs the effective prevention of insider threats from developing into major incidents when they are still in their infancy. Furthermore, the consideration of only rules and statistical thresholds does not allow the detection of abnormal activity in real time, which could be particularly problematic in dynamic environments where normal and abnormal patterns may change rapidly. Nevertheless, rule-based systems as well as statistical anomaly detection are based on a certain level and are insufficient for dealing with various types of behavioral patterns and have high rates of false positives.

2.3 The Role of Big Data Analytics in Enhancing Cybersecurity

Big Data Analytics is one of the main branches of cybersecurity in the modern world as it deals with the analysis of large amounts of data for the purpose of insider threat detection. This segment explores the enhancement offered by Big Data Analytics in cybersecurity using detailed procedures and cases.

Processing and Analyzing Vast Amounts of Data

Big Data Analytics is the procedure by which large data structures and unstructured data that cannot be managed by a conventional Analysis tool and technologies are examined. In cybersecurity, this capability is critical because it pertains to the various categories of data including log, network traffic, user, and system activity data. According to the study of (Luo et al. 2021) such data can be collected and trained into an analysis process in real-time, with the aim of identifying patterns and anomalies that possibly point to insider threats. Thus, big data processing is possible with the

help of methods such as parallel processing, distributed computing systems like Hadoop and Spark, as well as cloud-based analytical platforms.

Uncovering Hidden Patterns and Correlations
The other benefit of Big Data Analytics in cybersecurity is that, unlike the rest of the methods that analyze after the event, it has the capability of identifying unknown patterns in relations. The essence of the Big Data can also be analyzed via machine learning algorithms and statistical models which in turn assist firms in identifying the patterns of behaviors that are unlikely to occur due to normalcy, but which are executions of cybercriminals. The techniques include anomaly detection (Isolation Forest, One-Class SVM), clusterization, and predictive analysis (Kim et al. 2020). Thus, the application of these techniques helps security analysts to distinguish normal user behavior from insiders' malicious activity on the network. The significance of big data analytics for the cybersecurity context will be described by using the case studies that are provided below. For example, organizations track and analyze the user activity logs in real-time and the network traffic and data to detect unauthorized access or data loss. First, the comparison of multiple sources at the same time, for example, UBA data and EDR data will provide more information about the potential threat and prevent a reaction to the insider threat.

2.4 Leveraging Deep Learning Models for Anomaly Detection

Explanation of Deep Learning Models

Several studies have shown that Deep Learning, under a Neural Network framework has been the most efficient tool in anomaly detection in cybersecurity since it can deal with complex data and recognize elaborate patterns. According to (Sudar et al. 2021), the analysis indicates that Neural networks are much better positioned than conventional heuristic or statistical-based anomaly detection systems to train representations from raw data and are much more effective at identifying novel and complex insider threat behaviors. Neural networks consist of layers of nodes that are stacked and connected to learn a hierarchy of the features and are capable of performing non-linear mapping and feature detection which is not easily performed with other mathematics.

Applications in Anomaly Detection
Deep learning models are used in cyber security in different domains to identify suspicious clues of malicious insiders. For example, recurrent neural networks (RNNs) are good when it comes to analyzing the temporal patterns of user behaviors or network traffic typical behaviors (Al-Mhiqani et al. 2020). Convolutional neural networks or CNN are good at object detection particularly when it comes to anomaly detection like 'motion detection' when a person is seen using cameras or any detection of a bot in system log data.

Advantages Over Traditional Methods
Several benefits over conventional techniques can be attributed to deep learning in detecting elaborate patterns of insiders. First, it overcomes the limitations of having to update the detection rules every time there is a shift in the features and patterns of the data. Secondly, they can operate on high dimensions and unstructured data such as raw network packets or text logs thereby capturing all the relations and unusual behaviors. Finally, deep learning models can also offer real-time detection of anomalies and threats, which can be addressed even before they reach threatening

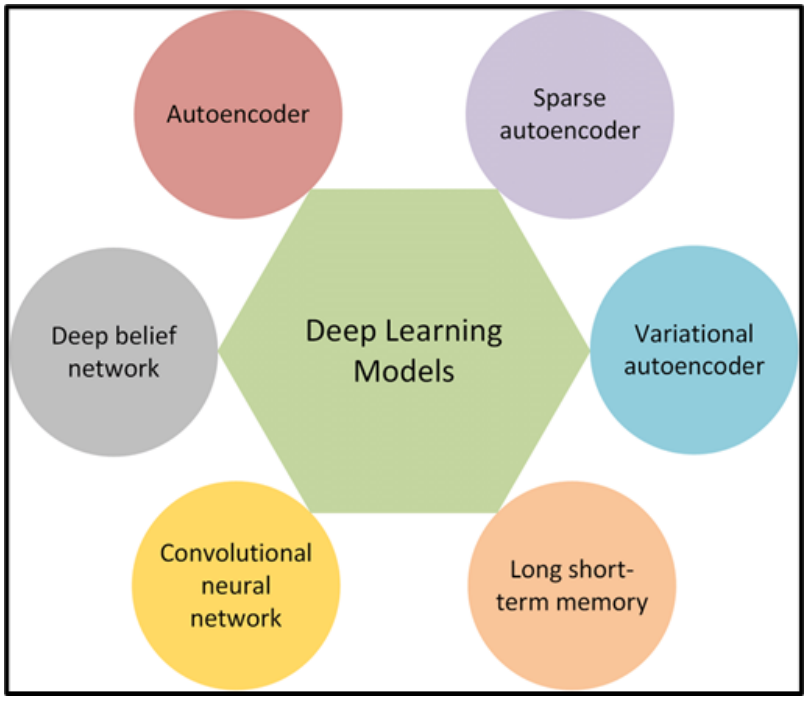


Figure 2.2: Deep learning models for detection (Al-amri et al. 2021)

proportions.

2.5 Integrating Big Data Analytics and Deep Learning for Enhanced Insider Threat Detection

Benefits and Challenges

Big data combined with deep learning can help to expand the possibilities of insider threat detection considerably, as the integration of the technologies shows important advantages. Big Data Analytics deals with larger datasets of mixed types that are ideal for feeding deep learning models that help identify the firm's structural patterns and unnoticeable irregularities (Al-Shehari and Alsowail, 2021). This integration improves threat detection capabilities since machine learning will be able to solve complex problems related to threats from data patterns. However challenges like integration of data, computational requirements for large data sets, and the need for expertise both in big data and deep learning act as impediments to implementation.

Architectural Frameworks and Models

There are several architectural constructs and patterns to overcome the challenges when incorporating Big Data Analytics and Deep Learning for insider threat detection. Solutions generally comprise big data processing frameworks such as Hadoop or Spark for different data types. Advanced Machine Learning algorithms like RNNs or deep autoencoders are utilized for end-to-end anomaly

detection across various forms of data including network logs, users, and system activities (Wategaonkar et al. 2024). These frameworks make it possible for data to be processed optimally and, equally as important, for deep learning models to process data in real-time or virtually real-time as it is necessary to be able to counter threats as soon as they arise.

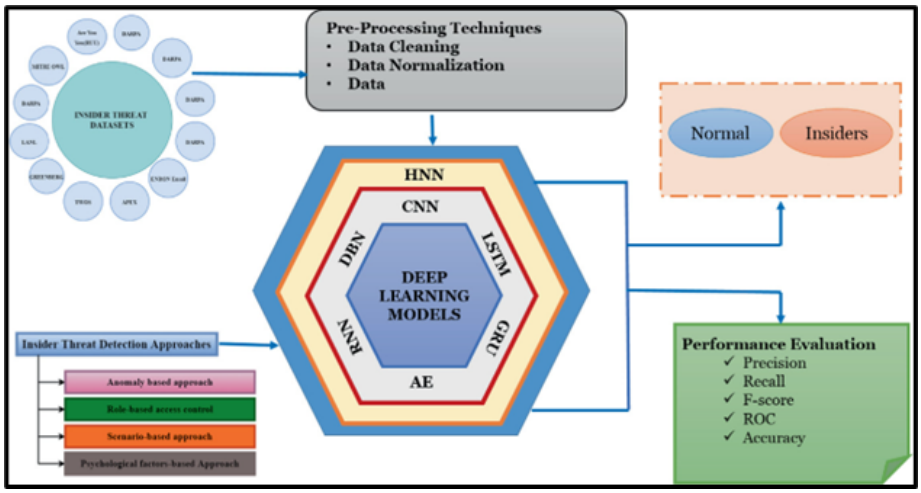


Figure 2.3: Overview of the models (Lavanya & Shankar Sriram 2022*a*)

Real-Time Processing and Adaptive Learning Features
Real-time processing capability is very important to enable immediate detection and handling of insider threats. Through the use of streaming data platforms, and distributed computing frameworks, organizations can perform analytics on data within streams in real time hence fixing anomalies faster. This is because Deep Learning provides significant elements of adaptability integral in updating the knowledge within the deployed models to accommodate new patterns and the dynamics in threat conduct. Such dynamic learning ability yields improved defense against new and advanced strategies of insider threats to protect organizational assets and sensitive information.

2.6 Current Challenges and Future Directions in Automated Analytic Tools for Insider Threat Detection

The following are some of the challenges that can hamper the implementation of automated analytic tools for insider threat detection. As per the views of (Demertzis et al. 2020), traditional tools have high false positive values, which overload security teams with unnecessary alarms and decrease productivity. Second, the variety of the data, both structured and unstructured, and the amount of them are essential and can become a problem, considering the difficulty of proper integration of multiple datasets into valuable information assets. In the same way, insider threat programs need tools that can be updated and reused in the light of new tactics in insider threat. Moreover, protecting identity and legal requirements while working with the concerned data increases the difficulty level of mere tool implementation. In response to these challenges, organizations are therefore looking at new solutions. Advanced machine learning algorithms that are specific to deep

learning models are promising in minimizing false positives because of the ability of algorithms to learn from past data and the current threat (Fotiadou et al. 2021). The incorporation of behavioral analysis and anomaly detection allows for distinguishing the subtle changes in the users' activities that signify internal threats. However, investments in Big Data platforms and cloud, analytical tools enable fast and capitalistic analyses of large datasets to boost the detection process.

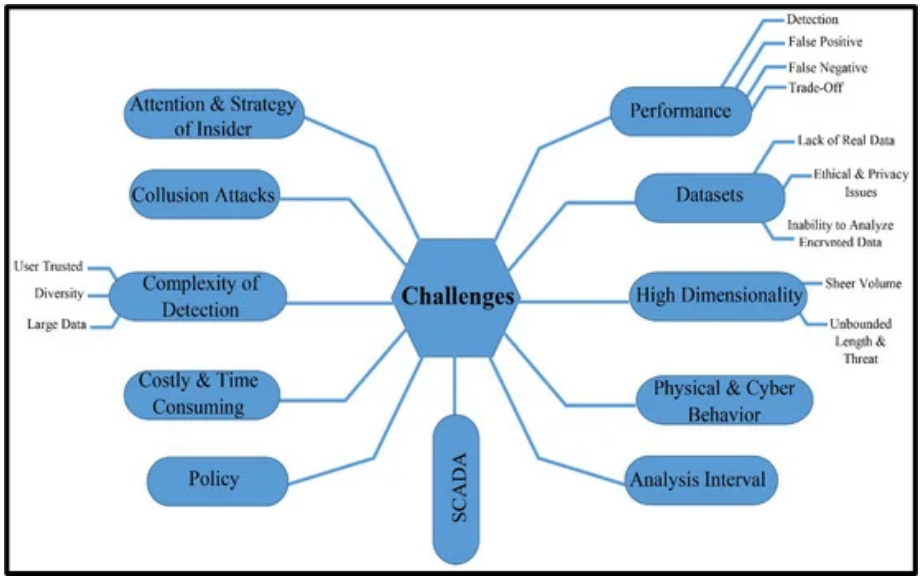


Figure 2.4: Challenges of Insider Threat Detection (Fotiadou et al. 2021)

Due to the dynamic changing nature of insider threats, there is always the need for further research and development of automated analytic tools. Future work should consider the optimization of threat detection by the contribution of big data analytics and the reinforcement of machine learning algorithms. Also, furthering the understanding and cohesiveness of analytic tools will enable the security teams to arrive at decisions as swiftly as possible. This recognition of privacy considerations and guidelines for data protection will be essential in creating efficient and ethical approaches for the identification of insider threats.

Insider Threat Insider threats, which are defined as any action made by an employee or contractor that may be detrimental to the business, such as sabotage or unapproved data removal, are a complicated and expanding problem for many organisations. Since insider attacks are less common than external threats, organisations typically focus on exterior dangers rather than insider threats (Gelles, 2016). Because traditional intrusion detection systems are not built or able to recognise malicious insider traffic, organisations typically invest less in an Insider Threat mitigation programme as part of a risk management strategy.

Importance of Insider Threat Detection
The detection of insider threats is paramount for organizations to safeguard their critical assets, intellectual property, and sensitive data from malicious insiders. Unlike external threats, insider threats pose a significant risk due to their proximity to the organization's systems and their knowledge of its operations. Moreover, insider threats can often evade traditional security measures, mak-

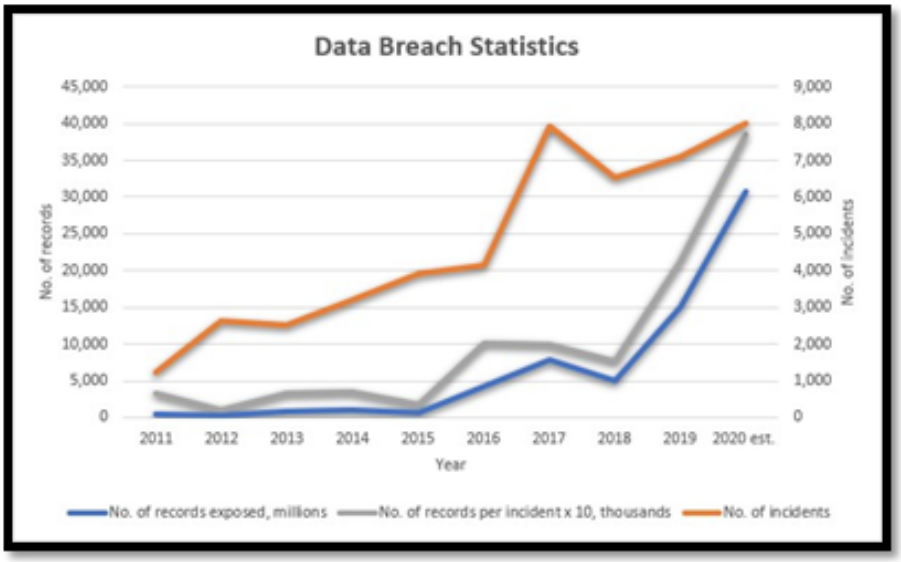


Figure 2.5: Reducing the Time it Takes to Detect Insider Breaches (Varma Vegesna 2022)

ing them difficult to detect using conventional approaches. Therefore, effective insider threat detection mechanisms are essential for early identification and mitigation of potential security breaches, thereby minimizing the risk of financial loss, reputational damage, and regulatory penalties.

Deep Learning in Anomaly Detection

Deep learning, especially neural networks, is a potent tool for anomaly detection across various domains like cybersecurity. Neural networks can learn intricate patterns from extensive data, enabling detection of subtle anomalies, such as those from insider threats. Unlike rule-based methods, deep learning automatically learns from raw data, capturing complex relationships effectively. Its ability to handle high-dimensional and diverse data makes it ideal for anomaly detection in complex organizational settings. In the context of insider threat detection, deep learning techniques offer several advantages, including:

1. Improved Detection Accuracy: Deep learning models can identify anomalous behavior with high accuracy, enabling organizations to distinguish genuine insider threats from benign anomalies.
2. Adaptability to Evolving Threats: Deep learning models can adapt to evolving insider threat tactics and strategies by continuously learning from new data, thereby enhancing the resilience of insider threat detection systems.
3. Scalability: Deep learning algorithms can handle large volumes of data efficiently, making them suitable for analyzing extensive datasets typically encountered in insider threat detection scenarios.
4. Real-time Detection: Deep learning models can perform anomaly detection in real-time, enabling organizations to detect insider threats promptly and respond swiftly to potential security incidents.

Major Problem Complexities

Anomaly detection deals with minority, unpredictable/uncertain, and infrequent events as opposed to majority, regular, or obvious patterns. This makes it different from other issues and tasks and presents certain special challenges to all (deep and shallow) detection methods:

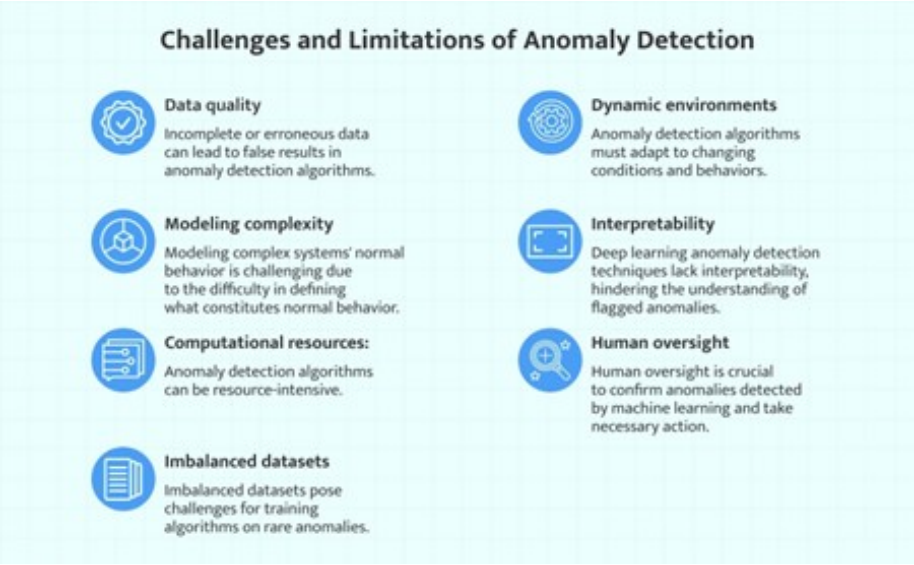


Figure 2.6: Challenges and Limitations of Anomaly Detection (Varun Chandola & Kumar 2009)

1. Lack of knowledge. Anomalies are linked to a multitude of unknowns, such as data structures, distributions, and examples with unknown abrupt behaviours. They are things like new terrorist attacks, fraud, and network intrusions that are unknown until they really happen.
2. Anomalies with heterogeneous classes. Due to the irregular nature of anomalies, two classes of anomalies may exhibit entirely distinct aberrant traits from one another. Robberies, car crashes, and burglaries, for instance, seem quite different visually on video surveillance.
3. Inequality of class and rarity. Unlike regular occurrences, which frequently make up the majority of the data, anomalies are usually isolated data events. It is therefore very difficult, if not impossible, to gather a significant number of labelled atypical events. As a result, large-scale labelled data is unavailable for most applications. Another reason for the class imbalance is that misclassifying anomalies typically entails far higher costs than misclassifying regular occurrences.
4. Various types of anomalies. Point anomalies, conditional anomalies, and group anomalies are the three different categories of anomalies that have been studied (Varun Chandola & Kumar 2009). Point anomalies are isolated occurrences that differ markedly from the majority, like a patient's aberrant health signs. Conditional anomalies, like abrupt temperature swings or fast credit card transactions, are peculiar data examples that only occur in certain settings. Group anomalies, such as dense subgraphs created by fictitious accounts in social networks, are subgroups of data examples that collectively deviate from the norm.

Main Challenges Tackled by Deep Anomaly Detection
There are several detecting issues as a result of the aforesaid complicated problem nature. While certain issues, such scalability in relation to data size, have been successfully resolved recently, there are still other issues that need to be resolved, for which deep anomaly detection might be crucial.

5. Low recall rate for anomaly detection in CH1. Identifying every anomaly is challenging since they are so uncommon and varied. While many commonplace occurrences are mistakenly reported as anomalies, actual yet complex abnormalities go unnoticed. The state-of-the-art anomaly detection techniques, particularly the unsupervised techniques (Liu et al. 2012), continue to have a high false positive rate on real- world datasets (Campos et al. 2016) despite the introduction of several techniques over the years. One of the most crucial and challenging problems is figuring out how to increase detection recall rates while decreasing false positives. This is especially true given the high cost of missing anomalies.

6. Finding anomalies in high-dimensional, non-independent data in CH2.In a low- dimensional space, anomalies frequently show clear anomalous traits, but in a high-dimensional space, they tend to disappear into the background. The identification of high-dimensional anomalies has long been a challenge (Zimek et al. 2012). A simple option is to perform anomaly detection in a reduced lower-dimensional space that is covered by a limited fraction of original features or newly generated features, as in feature selection-based techniques (Pang et al. 2019) and subspace-based methods (Keller et al. 2012). Finding complex feature interactions and couplings (such as those that are high-order, nonlin- ear, and heterogeneous) may be crucial in high-dimensional data, but anomaly detection still faces significant challenges in this area.Furthermore, successful anomaly detection downstream depends on how to ensure the new feature space maintained adequate information for particular detection techniques. This is difficult because of the previously described unknowns and heterogeneity of anomalies. Furthermore, it might be difficult to identify abnormalities from examples that can be interdependent, whether through temporal, geographical, graph-based, or other linkages [(Aggarwal 2016),(Akoglu et al. 2014),(Gupta et al. 2014)].

7. Data-Efficient Learning of Normality/Abnormality in CH3.The problem of limited availability of labelled anomaly data is addressed by data-efficient learning of normalcy and abnormality. Extensive labelled data is necessary for traditional supervised anomaly detec- tion approaches, which is frequently unfeasible. Unsupervised techniques rely on assumptions about anomaly distributions and lack prior knowledge of anomalies. Using easily accessible labelled data, semi-supervised and weakly supervised anomaly detection techniques acquire expressive representations of normalcy and abnormality (Aggarwal 2016). While weakly supervised techniques deal with partial, erroneous, or incomplete anomaly labelling, semi- supervised techniques make use of labelled normal data. Learning expressive representations with limited labelled anomaly data and adapting detection algorithms to new abnormalities are the main problems.

8. Anomaly detection with resilience to noise in CH4. Numerous techniques for weakly or semi-supervised anomaly detection use the assumption that the labelled training data are clean, which leaves open the possibility of noisy examples being incorrectly labelled as belonging to the opposite class. In these situations, we may resort to using unsupervised techniques, but this does not make use of the actual labelled data. Furthermore, large-scale anomaly-

contaminated unlabeled data are frequently present. These unlabeled data may be utilised by noise-resilient algorithms to improve detection accuracy. So, either incorrectly labelled data or unlabeled abnormalities might be the source of this noise. The primary difficulty lies in the fact that noise levels can vary greatly across datasets and that noisy cases might be dispersed unevenly throughout the data space.

9. Complex anomaly detection in CH5. Since conditional anomalies and group anomalies behave very differently from point anomalies, the majority of methods now in use are only suitable for handling point anomalies. Including the idea of conditional or group anomalies in anomaly measurements or models is one of the primary challenges here. Furthermore, many applications need the identification of anomalies using numerous heterogeneous data sources, such as multidimensional data, graphs, images, text, and audio data (Pang et al. 2019). Current approaches primarily concentrate on detecting anomalies from single data sources. A primary obstacle is that certain irregularities are only discernible when examining two or more data sources.

10. Interpretation of anomaly in CH6.Anomalous detection models that are utilised as black-box models directly may pose significant dangers in many safety-critical fields. A possible algorithmic bias against the minority groups shown in the data, such as underrepresented groups in fraud and crime detection systems, might result, for instance, from the infrequent data examples reported as anomalies. A good way to reduce this kind of risk is to establish algorithms for explaining anomalies that give clear indications as to why a particular data instance is considered anomalous. After that, human specialists can investigate and address the prejudice. In certain cases, giving such an explanation may be just as crucial as detecting accuracy.Nevertheless, the majority of anomaly detection research ignore the potential to explain the abnormalities that are detected in favour of concentrating solely on detection accuracy. Creating anomaly detection models that are naturally comprehensible is also essential, but striking a balance between the model's efficacy and interpretability continues to be a major obstacle (Pang et al. 2019).

Big Data Analytics for Insider Threat Detection

In the digitally linked world of today, almost every action and contact produces data. A plethora of sources, including online logs, cellphones, social media, satellite photos, human genomes, consumer transactions, astronomical records, and biological databases, contribute to this enormous and varied flood of data, which is commonly referred to as "Big Data." Researchers and analysts have both enormous potential and formidable obstacles due to the sheer amount, velocity, and variety of this data. Key Characteristics of Big Data:

1. Volume: TLarge volumes of data are produced; they are frequently expressed in terabytes or petabytes. Large datasets require different tools for data processing than those found in traditional technologies.

2. Velocity: Due to the rapid generation of data, processing must be done in real-time or very near real-time in order to provide timely insights.

3. Variety: Unstructured, semi-structured, and structured data are among the several formats that data may take. Audio, video, text, pictures, and sensor data are a few examples.

Figure 2.7: 5 V's of Big Data (Guntur 2015)

4. Veracity: Strong techniques for data cleansing and validation are required since data might differ in quality and accuracy.
5. Value: The ultimate objective of big data analytics is to mine rich insights that stimulate creativity and help make decisions.

Role of Big Data in Insider Threat Detection
Due to its ability to handle, analyse, and understand massive amounts of data produced by organisational systems, big data analytics plays a critical role in the detection of insider threats. It can be challenging to identify insider threats using conventional techniques since they frequently leave behind minute evidence in data, such as user activity logs, network traffic, and system event logs. Organisations may easily discover abnormal behaviours suggestive of insider threats by utilising big data analytics to combine and correlate data from many sources. Big data analytics solutions can provide real-time monitoring features that let businesses quickly identify and address insider risks.

Techniques for Analyzing Big Data
Several techniques are employed in analyzing big data for insider threat detection, including:

1. Data Preprocessing: Data preprocessing involves cleaning, filtering, and transforming raw data to make it suitable for analysis. This may include removing noise, handling missing values, and standardizing data formats to ensure consistency and accuracy in analysis.
2. Rule-Based Detection: This method uses predefined rules to identify potential threats. These rules are typically based on known patterns of malicious behavior. The effectiveness of this technique heavily depends on the quality and comprehensiveness of the rules established.
3. Anomaly-Based Detection: This involves identifying deviations from normal behavior. Anomaly detection is implemented in several phases:
   LSTM Prediction: Using Long Short-Term Memory (LSTM) networks to predict future data points based on historical data. This method helps in detecting anomalies by comparing predicted values with actual values.
   Error Vector Computation: Calculating the difference between predicted and actual values to generate an error vector. Anomaly Detection: Using the Mahalanobis distance to determine whether the error vec- tors represent anomalies.
4. Deep Anomaly Detection Techniques:
   Unsupervised Deep Anomaly Detection (DAD): Techniques such as Auto-Encoders, Long Short-Term Memory Networks, and Generative Adversarial Networks (GANs) are employed. These techniques do not require labeled data and can learn the inherent characteristics

of the dataset to identify outliers.
Semi-Supervised Deep Anomaly Detection: This approach assumes that all training data belongs to one class, making it easier to identify anomalies. Techniques such as Restricted Boltzmann Machines, Auto-Encoders, and Deep Belief Networks are used in this category.

5. Time-Series Analysis: For analyzing time-series big data, techniques are adjusted to handle the unique characteristics of temporal data. This includes both uni-variate and multi-variate time-series anomaly detection, which takes into account the sequential nature of the data.

Anomaly Detection in Insider Threats
Traditional Approaches to Anomaly Detection : Traditional approaches to anomaly detection in the context of insider threats often rely on rule-based systems, statistical methods, and signature-based approaches. These methods typically involve the following techniques:

1. Rule-based Systems: Rule-based systems define predefined rules or thresholds based on known patterns or behaviors to identify anomalies. For instance, rules may be established to flag unusual access patterns, such as accessing sensitive files outside of business hours or from unauthorized locations.

2. Statistical Methods: Statistical anomaly detection techniques analyze historical data to identify deviations from normal behavior. Methods such as mean/variance analysis, clustering, and time-series analysis can help detect outliers or anomalies in user activity logs, network traffic, or system logs.

3. Signature-based Approaches: Signature-based anomaly detection relies on predefined signatures or patterns of known attacks or malicious behaviors to identify anomalies. For example, intrusion detection systems (IDS) use signatures of known malware or attack patterns to detect unauthorized access attempts or suspicious network traffic. While traditional approaches have been effective in certain scenarios, they have limitations in de- tecting insider threats, such as their reliance on predefined rules or signatures, inability to adapt to evolving attack strategies, and high false positive rates.

Deep Learning Techniques for Anomaly Detection
Deep learning techniques offer a promising approach to anomaly detection in insider threat scenarios due to their ability to automatically learn complex patterns and representations from data. Some of the commonly used deep learning techniques for anomaly detection include:

4. Autoencoders: Autoencoders are neural network architectures designed to learn efficient representations of input data by compressing and then reconstructing it. Anomalies are detected based on the reconstruction error, where higher errors indicate deviations from normal behavior.

5. Recurrent Neural Networks (RNNs): RNNs are capable of capturing temporal dependencies in sequential data, making them suitable for analyzing time-series data, such as user activity logs or network traffic. RNNs can detect anomalies by learning patterns in sequential data and identifying deviations from expected behavior.

6. Generative Adversarial Networks (GANs): GANs consist of two neural networks, a generator and a discriminator, trained simultaneously to generate realistic data samples and distinguish

between real and fake samples. GANs can be used for anomaly detection by generating synthetic data and comparing it with real data, identifying discrepancies indicative of anomalies. Deep learning techniques offer several advantages for anomaly detection in insider threat scenarios, including their ability to learn complex patterns, adapt to evolving threats, and handle high- dimensional and heterogeneous data effectively.

Performance Evaluation of Anomaly Detection Methods
Performance evaluation of anomaly detection methods in insider threat scenarios involves assessing their effectiveness in detecting anomalies while minimizing false positives and false negatives. Common metrics used for performance evaluation include:

7. Detection Rate: The proportion of true anomalies correctly identified by the detection system.
8. False Positive Rate: The rate of false alarms or benign instances incorrectly flagged as anomalies.
9. Precision and Recall: Precision measures the proportion of true positives among all instances flagged as anomalies, while recall measures the proportion of true anomalies detected by the system.
10. F1 Score: The harmonic mean of precision and recall, providing a balanced measure of the detection system's performance. Performance evaluation may involve using labeled datasets with known ground truth labels to assess the detection system's accuracy, as well as conducting real-world testing to evaluate its effectiveness in detecting insider threats in operational environments.

2.7 Literature Gap

Despite recent developments in automated analytic tools for insider threat detection, there are some research gaps in the current body of knowledge. The first significant area pertains to the question of how heterogeneous sources of data can be smoothly incorporated into analytic paradigms. It must be noted, that research articles analyze Big Data Analytics and Deep Learning as two distinct approaches to address insider threat, however, there are few studies that attempt to provide an integrated solution of both the aforesaid methodologies. Also, it is still difficult to set norms and indicators for measuring the performance of automated analytical tools with a fair degree of precision. There is often a lack of methodological consistency for evaluating the false positive rates, the accuracy of detection, and scalability in different organizational contexts and threats. This circumstance prevents the comparison of different types of approaches for detection and reduces the possibility of generalization. Also, insider threats are unpredictable making it necessary to constantly update the detection methods. There are, however, few scientific studies focusing on the real-time adaptation of the organization's protection capabilities, as well as its ability to detect changing insider threat behaviors. Insider threat research in the future must seek to fill these gaps by designing sound frameworks to incorporate more advanced technology and smart learning capabilities to cope effectively with the dynamic forms of threat as noted in today's complex threat environment.

2.8 Conceptual Framework

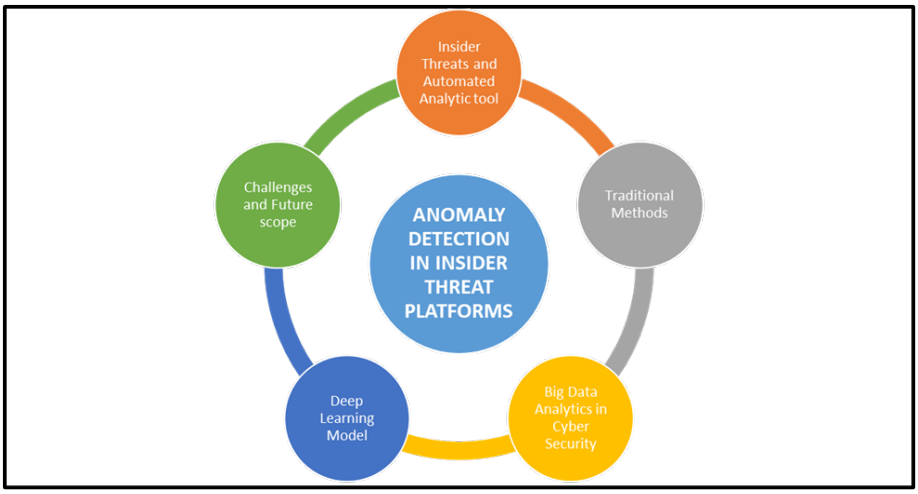


Figure 2.8: Conceptual Framework

2.9 Summary

This chapter presented an analysis of the current resources for automating insider threat analysis with drawbacks such as inflated false positive rates and data fusion difficulties identified. Big Data Analytics and advancements in machine learning present promising directions, but the current lack of integration and measures of evaluation remain. Future studies should continue exploring ways to improve real-time adaptability and implement reference models to further refine insider threat detection paradigms.

Chapter 3

Research Approach and Methodology

3.1 Introduction

In the methodology chapter, the approach, design, data collection, analysis techniques, evaluation criteria, tools used, and ethical considerations used in establishing Automated Analytic Tools for the Identification of Insider Threats are documented. This chapter explains how deep learning techniques and big data analysis enable the improvement of the detection rate and management of insider threats in cybersecurity. Ethical issues are crucial to protect the data and prevent misuse of the application, and measures demonstrate the effectiveness of the tool for real-time anomaly detection.

3.2 Research Approach

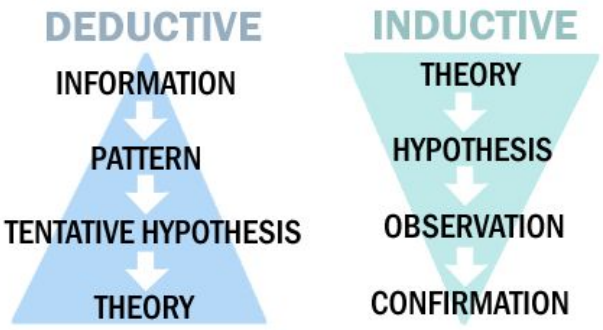


Figure 3.1: Overview of Research Approaches (*Research Techniques Deductive and inductive research* 2024)

This research has used inductive research to analyze the given data and develop an intelligent analysis system to identify insider threats. Inductive reasoning can be defined as an approach to reasoning that involves particular data and observations with the purpose of developing broad theories and hypotheses. Therefore, the application of the inductive approach could be justified to some extent in terms of this research, as this approach provides a set of effective tools for systematic analysis of the existing literature and experience in the detection of insider threats by means of automatic analysis.

Justification of the Choice
As for the method of inductive reasoning, the rationale for it is mainly based on the fact that the present study can be considered as the first step in the investigation of the research problem, that is, the study can be viewed as exploratory research. This makes it possible to integrate the outcomes outlined in the literature and produce novel propositions and discussions concerning potential ways for addressing and reducing ITP with the help of automated analytical tools (Al-Mhiqani et al. 2020). In addition, real-time anomaly detection for pattern and tendency recognition, which is vital to counter the novel tactics exploited by the insiders is also based on inductive reasoning. This methodological decision enriches the results with significance and co-creates the cybersecurity interior threat detection theories in the best way.

3.3 Research Design

This research work has used exploratory research design to assess the extent of the use of Deep Learning and Big Data Analytics in the detection of insider security threats. Exploratory research is used due to its effectiveness in the recognition of new areas for investigation and the collection of details in areas that may not be explored much (Gayathri et al. 2020). Therefore, the discussed design allows for understanding the benefits and possibilities of using the described innovative approaches to improve cybersecurity against IT insiders' misconduct in an organization.

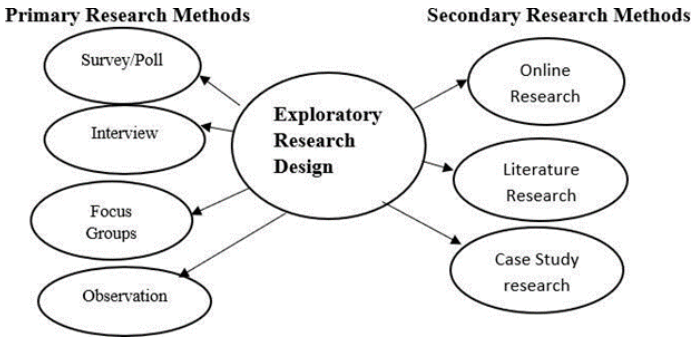


Figure 3.2: Overview of Exploratory Research Design (Saka et al. 2023)

Justification of the Choice
The reason for using an exploratory research design is the desirability of identifying the current and future ways of detecting insider threats. The design allows the analysis of unseen patterns in the data and evaluating the tentative solutions on increasing the detection ratio and decreasing

the false-positive ratio for the given data type along with multiple machine learning algorithms and online actions. Also, exploratory research allows the changes and adjustments of the research approach that, along with findings, and achievements continue right through the process. This means that the process is iterative because of the constantly dynamic nature of insider threats and ensures that the developed analytic tool for insider threats handles current cybersecurity issues well.

3.4 Data collection

The present research relies on secondary research data obtained from Kaggle, an online source of big data sets. The datasets are chosen from Kaggle's extensive resource for cybersecurity, specifically for identifying cycles in ransomware and zero-day attacks, which is cyclostationary (Islam et al. 2021). This dataset consists of timestamps in case of having a record of attack time, a flag indicating the type of attack, a protocol in order to analyze how an attack transpires, network flow where one can observe how data is flowing, and lastly ransomware family. Secondary data collection is opted for given the time-saving and cost-saving isolation it has in reaching out for huge volumes of existing data that are relevant to the research question of the study.

Justification of the Choice
The use of secondary data collection on Kaggle is further supported by the fact that the Kaggle platform hosts well-formatted and high-quality data shared by data scientists and researchers across the world. These datasets are usually well-documented and well-curated, and in many cases, the data is cleaned, preprocessed, and made available for analysis saving time. Furthermore, secondary data from Kaggle makes the researchers concentrate more on the analysis and experimentation thus enhancing the development and validation of the automated analytic tool for insider threat detection.

3.5 Data Analysis

This paper uses Convolutional Neural Networks (CNNs) and classifier models for the analysis of datasets from Kaggle. CNNs are utilized due to their effectiveness in modeling spatial relations between data points, whereas, the process of quantizing patterns and relations extracted from networks' activities and user log data for modeling insider threat detection requires explicit quantization. The classifier models involved in the system and deep learning, in particular, are used to detect and differentiate the unusual activities common with insiders (Elmrabit et al. 2020). Such models are used on the labeled dataset to distinguish between normal and suspicious activity to lower the false positive and false negative rates.
Normalization is the last step in data pre-processing in which data is adjusted for modeling by removing aspects that may embarrass the model in its operation. Therefore, there are various performance indices including accuracy, precision, recall, and F1-score useful in evaluating the performance of the approaches and the CNNs in real-time insider threat detection contexts. The goals of the following research are to show how the CNNs and classifier models may be beneficial to cybersecurity solutions for insider threats through deep learning paired with big data.

3.6 Evaluation Metrics

There are various methods that are employed in establishing the effectiveness of the automated analytic tool for identifying insider threats, which are discussed in the section below. In general, accuracy depicts the total amount of correctly classified instances in the model and this term describes the number of accurate classifications. It measures the correct positive prediction to the total positive prediction ratio with much regard to the number of false alarms that the tool can produce. Recall, or specificity, refers to the number of selective true positives regarding the total number of actual positives, which underlines the effectiveness of the tool in identifying insider threats (Ferrag et al. 2021). This is good for a balanced evaluation of the tool's performance in terms of precision and recall and is expressed as the F1-score which is an average of the precision and recall in their harmonic sense. All these metrics help in a way that the tool is assessed on its ability to detect abnormalities in real-time and also accommodate new tactics resulting from the insiders and, therefore, validate the importance of the tool in enhancing security.

3.7 Tools and Techniques

Tools Used
The research involves the use of Python programming language further augmented by TensorFlow and Keras to develop deep learning tools including Convolutional Neural Networks (CNNs) and classifiers. These tools help in data pre-processing, training, and testing of the model which is required for the construction of the analytic tool for the identification of insider threats (Rosa et al. 2021).
Techniques Employed
Data normalization, feature extraction, and using the convolutional neural network and classifier models are some of the techniques used. These techniques are therefore relevant when it comes to pattern analysis, especially in the logs of the network and user behavior data obtained from Kaggle.

3.8 Ethical consideration

Specifically, the ethical concerns in the present study are mainly associated with data protection, individuals' anonymity, and proper employment of AI applications in the context of cybersecurity. Anonymized datasets from Kaggle also contain only datasets that do not violate privacy rights or laws regarding the confidentiality of paper or electronic information or records (Gupta et al. 2020). Also, it practices ethics in dealing with data by affirming close documentation of its data source and the process used to deal with the data. Such openness ensures that the work done conforms to global best practices with the results derived from the research can be verified by other researchers together with members in the cybersecurity field. Also, the use of artificial intelligence in different branches of medicine, including deep learning models and classifier algorithms, requires the reduction of prejudice and fairness in the decisions made by the designed model (Sheykhkanloo & Hall 2020). A lot of attention is paid to the consequences of false positives and false negatives when it comes to observing insiders, trying to avoid all the risks that come with incorrect classification. The ethical consideration therefore embraces the principles of research advocating for the rights of people's privacy and disseminating research findings to improve Cybersecurity practices appropriately.

3.9 Summary

The chapter on methodology gives an idea of how to build an automated analytic tool for insider threat detection. It entails an exploratory research design that involves the use of secondary data from Kaggle, and analysis through CNNs and other classifiers. The overall performance of the tool is analyzed by using evaluation metrics like accuracy, precision, recall, and F1-score. It is important to note that ethical considerations with regard to the application of AI include data protection and the right use of the technology. Thus, the proposed approach could be considered as a perspective for enhancing cybersecurity with deep learning and big data methods to counter emerging insider threats.

Chapter 4

Analysis of Secondary Data

4.1 Introduction

The results section reports the findings of the study and discusses the capabilities of deep learning and big data analytics for insider threat systems' anomaly detection. It starts with an assessment of the data and covers all the procedures necessary to build a model. The chapter then goes on to systematically demonstrate the benchmark of performance across different ML algorithms such as Random Forest, KNN, Gaussian NB, MLP, Decision Tree, and CNN. This chapter assesses the ability of the models to detect insider threats using a wide range of visualizations and comparisons, which can be the basis for the subsequent discussion of the results' significance.

4.2 Results analysis

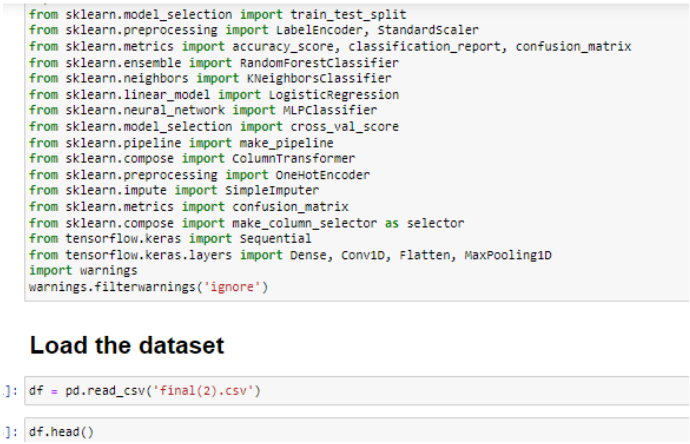
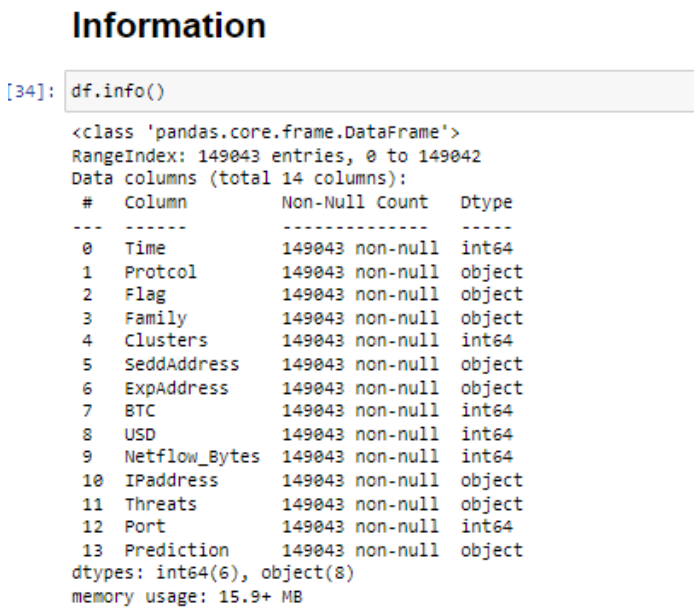


Figure 4.1: Importing libraries and dataset (Source: Developed in Google Collab)

This figure shows the initial process of establishing the environment for the analysis including the importation of some important packages in Python. The libraries have included pandas for data analysis, NumPy for numerical computations, Matplotlib for graphical presentation of data, scikit-learn and TensorFlow for machine learning model development and assessment. The data collected in the previous step is then imported into a pandas dataframe, which is the first step of feature engineering. These steps form the theoretical background that is essential for further data analysis and model building in the field of insider threat identification.



Information

```
[34]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149043 entries, 0 to 149042
Data columns (total 14 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Time           149043 non-null  int64
 1   Protcol        149043 non-null  object
 2   Flag           149043 non-null  object
 3   Family         149043 non-null  object
 4   Clusters       149043 non-null  int64
 5   SeddAddress    149043 non-null  object
 6   ExpAddress     149043 non-null  object
 7   BTC            149043 non-null  int64
 8   USD            149043 non-null  int64
 9   Netflow_Bytes  149043 non-null  int64
 10  IPaddress      149043 non-null  object
 11  Threats        149043 non-null  object
 12  Port           149043 non-null  int64
 13  Prediction     149043 non-null  object
dtypes: int64(6), object(8)
memory usage: 15.9+ MB
```

Figure 4.2: Dataset Information Result (Source: Developed in Google Collab)

This figure also shows the general layout of the dataset for a quick reference to some of the general features like the number of rows, the nature of data displayed and the type of data contained in each column. The dataset has 149,043 records with 14 columns, out of which some are of int64 type while other entries are of object type. These are the different columns containing information that may be important in identifying insider threats, such as time, protocol, flags, family, predictions, and others.

```
df.describe()
```

	Time	Clusters	BTC	USD	Netflow_Bytes	Port
count	149043.000000	149043.000000	149043.000000	149043.000000	149043.000000	149043.000000
mean	21.466979	2.349295	30.554605	14863.441114	2021.278651	5063.949967
std	15.883598	2.828759	101.447102	26849.434659	2271.420987	2.696011
min	-10.000000	1.000000	1.000000	1.000000	1.000000	5061.000000
25%	8.000000	1.000000	8.000000	512.000000	353.000000	5062.000000
50%	19.000000	1.000000	13.000000	4321.000000	1031.000000	5062.000000
75%	32.000000	2.000000	22.000000	18454.000000	3188.000000	5066.000000
max	96.000000	12.000000	1980.000000	126379.000000	12360.000000	5068.000000

Figure 4.3: Summary Statistics (Source: Developed in Google Collab)

The figure below shows the summary statistics of the variables, providing a clear description of the numerical variables such as *Time*, *Clusters*, *BTC*, *USD*, *Netflow_Bytes*, and *Port*. The descriptive results include the count, mean, standard deviation (SD), minimum (min), and maximum (max) values for all variables, as well as the 25th, 50th, and 75th percentiles.

For example, the mean of the converted *USD* is 14863.441, with values ranging from 1 to 125379, signifying variability in the findings. These statistics provide insights into the means, variance, and possible skewness of these variables, which are crucial for subsequent data analysis and modeling by product.

Checking null values

```
df.isnull().sum()
```

```
Time            0
Protcol         0
Flag            0
Family          0
Clusters        0
SeddAddress     0
ExpAddress      0
BTC             0
USD             0
Netflow_Bytes   0
IPaddress       0
Threats         0
Port            0
Prediction      0
dtype: int64
```

Figure 4.4: Checking null values (Source: Developed in Google Collab)

This figure illustrates an analysis of whether or not the dataset contains null values, which is important in advanced analytics. The output shows that there is no missing value for any of the

14 columns available in this dataset. The lack of null values is important as it can be an indication that the dataset is complete with no missing values thus no data imputation or data cleaning will be needed. This also helps in doing the analysis and training of the model without feeling that some data is missing in the data set.

Detect and remove outliers using Z-score

```
]: from scipy.stats import zscore
   numeric_cols = df.select_dtypes(include=[np.number]).columns
   z_scores = np.abs(df[numeric_cols].apply(zscore))
   df_cleaned = df[(z_scores < 3).all(axis=1)]
   print("Original dataset size: ", df.shape)
   print("Dataset size after outlier removal: ", df_cleaned.shape)

   Original dataset size:  (149043, 14)
   Dataset size after outlier removal:  (132305, 14)
```

Figure 4.5: Detecting outliers (Source: Developed in Google Collab)

The above figure illustrates how outliers are identified and excluded from the dataset to help clean the data using the Z-score method. The numbers in the columns have been analyzed with the Z-score, which indicates data that deviates from the mean by more than three standard deviations. Thus, it is observed that the number of entries is 132305 after eliminating all the outlier components, which was originally 149,043 rows of data, proving that there are many outliers there which have been excluded. The process of removing these outliers is important in cleaning the data that is to be fed into these machine-learning models. It enhances the credibility of the models because the models incorporate many factors and decision-making mechanisms that can be distorted by the outliers.

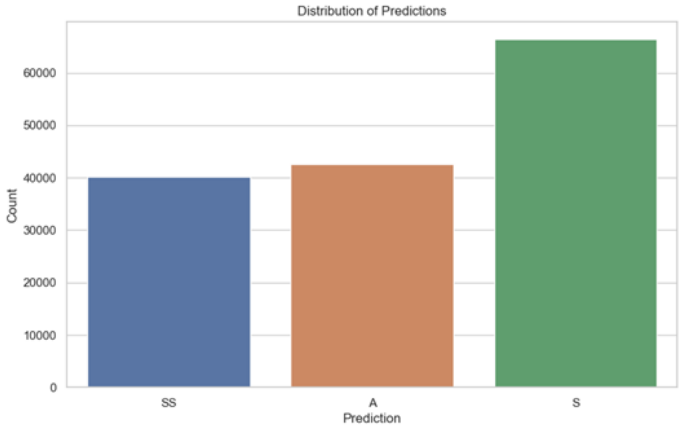


Figure 4.6: Distribution of Predictions (Source: Developed in Google Collab)

This figure shows how many instances there are, and what is the predicted value of each instance in the data set. The horizontal axis indicates the various prediction types while the vertical axis gives the number of occurrences corresponding to each type. The plot identified that category 'S' has the highest count on the y-axis which means this is the most frequent outcome in the dataset.

It also assists in identifying the dispersion of classes in a dataset and this is an essential element in evaluating the reputed accuracy of any predictive model.

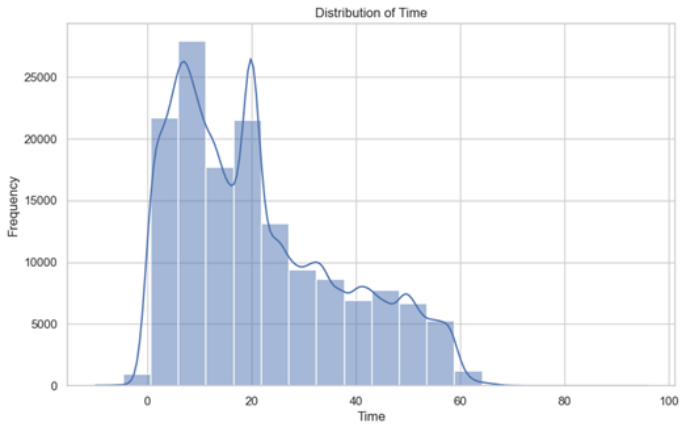


Figure 4.7: Distribution of Time (Source: Developed in Google Collab)

This figure illustrates the time series plot of the relative frequency of the Time variable in all values of the data set while forming a histogram. There is also a kernel density estimate (KDE) curve to supplement the histogram to provide a smooth curve approximation of the probability density function of the data. The values in the x-axis are the Time values and the y-axis represents the occurrence of evaluated events. The outlier can be observed in the plot for variable labelling Time where the range of minimum and maximum values of the variable plotting is large and most of the points are in the midline or around this line. This means that looking at the multiple peaks, there are subtle patterns or groups of data that are not easily recognizable.

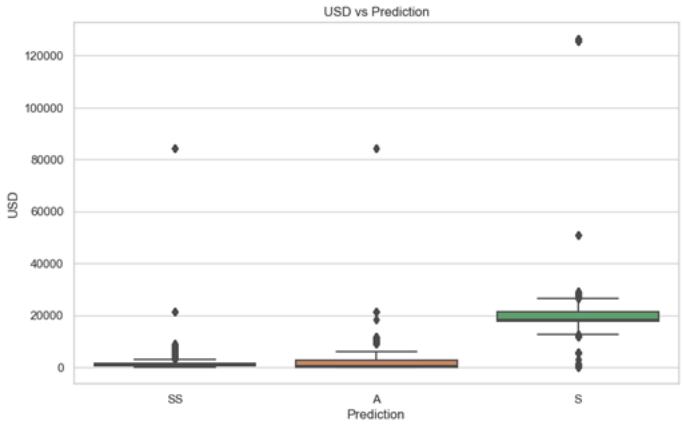


Figure 4.8: USD vs Prediction visualization (Source: Developed in Google Collab)

This boxplot has been used to present the USD values with the support of the Prediction categories. The horizontal axis informs who makes the predictions, and the vertical axis is the stage for the

USD value. Each of the box plots offers information on the median and interquartile range of USD within the prediction type, as well as the whiskers signifying a possibility of outliers. This plot is also useful in demonstrating the variation of USD values within such predictions to measure how each category has a different monetary value. This is particularly helpful in giving a visual representation of the USD values that are associated with these predictions.

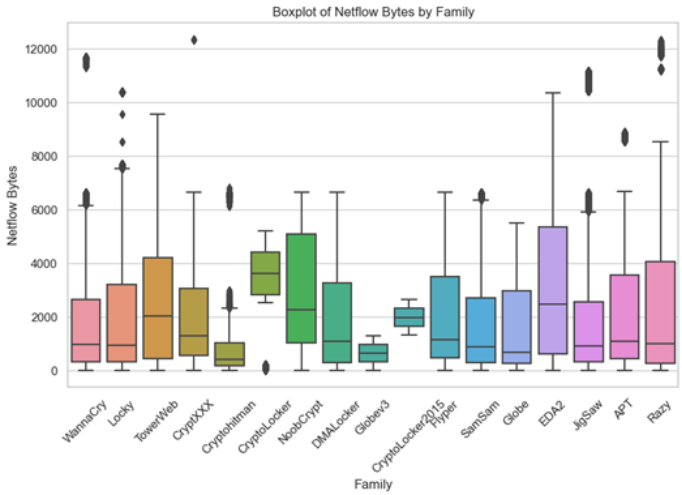


Figure 4.9: Boxplot of Netflow Bytes by Family (Source: Developed in Google Collab)

The description of the box plot has highlighted the Netflow bytes by family and the horizontal axis has presented the category of family along with the vertical axis presenting the value of netflow bytes. The boxplots represent Netflow Bytes for each Family group and present the median, quartiles, and outliers. The plot offers information regarding how Netflow Bytes is dispersed among families, the location of the median value and the difference in range. This visualization helps determine if a particular family has a high or a low netflow activity which is essential in studying the traffic flow.

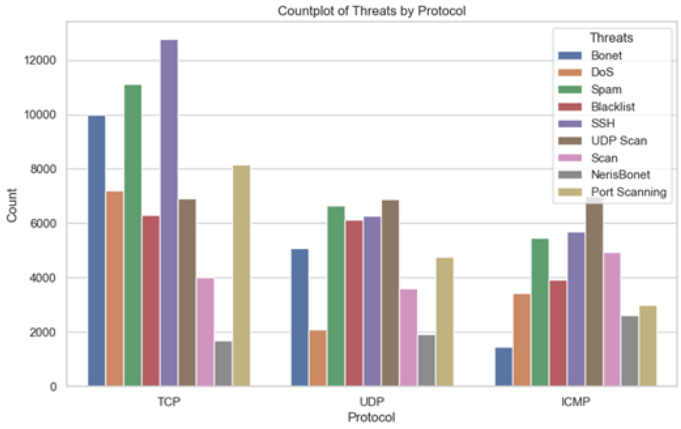


Figure 4.10: Countplot of Threats by Protocol (Source: Developed in Google Collab)

This figure shows a countplot which helps to understand how many Threats belong to a specific protocol category. The x-axis indicates each protocol while the y-axis gives the number of threats in each type. The hue parameter enhances the Threats in distinguishing colour to each protocol category for a colour-coded presentation. This plot displays how threats are allocated across the various protocols, and whether these threats have regularities. These insights are beneficial to draw attention to the fact that specified risks are connected to particular protocols and help implement corresponding security measures and enhance the strategies regarding Insider Threats' identification.

Encoding categorical variables

```
5]: from sklearn.preprocessing import StandardScaler, LabelEncoder
    label_encoders = {}
    for column in ['Protcol', 'Flag', 'Family', 'SeddAddress', 'ExpAddress', 'IPaddress', 'Threats', 'Prediction']:
        label_encoders[column] = LabelEncoder()
        df_cleaned[column] = label_encoders[column].fit_transform(df_cleaned[column])
```

Prepare data for machine learning models

```
6]: X = df_cleaned.drop(['Prediction'], axis=1)
    y = df_cleaned['Prediction']
```

Split data into train and test sets

```
7]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Standardizing the data

```
8]: scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

Figure 4.11: Data Preparation (Source: Developed in Google Collab)

This figure also shows the creation process of the new numerical variable, which requires categorical variables for encoding and preparation of data for feed to models. The Protocol, Flag, and Family columns which are categorical variables are encoded using 'LabelEncoder' to make them a format that can be processed by a model. The data is then divided into two different sets in an 80-20 ratio and it has indicated that 80% of the variables are present in the training set and the rest of the

variables are available in the testing set. In order to bring uniformity and normalize the feature space, "StandardScaler" is used so the features follow a mean of 0 and a standard deviation of 1 across the entire feature space. It is necessary for enhancing the performance of a model and helps in getting the right scale of features to achieve accurate and efficient training and testing of the model.

```
print("Confusion Matrix:\n", rf_conf_matrix)
```

```
Random Forest Accuracy: 0.9946714031971581
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7661
           1       0.99      0.99      0.99     11286
           2       1.00      1.00      1.00      7514

    accuracy                           0.99     26461
   macro avg       0.99      0.99      0.99     26461
weighted avg       0.99      0.99      0.99     26461

Confusion Matrix:
 [[ 7612    39    10]
 [   48 11215    23]
 [    0    21  7493]]
```

Figure 4.12: Random Forest Classification (Source: Developed in Google Collab)

The accuracy obtained on the test set by the Random Forest Classifier is quite high which is equal to 99.47%. The classification report shows consistently high values for precision, recall, and F1-score for all classes with a macro-average of 0.99 for precision, recall, and F1-score. Class 2 produced zero false negative results and zero false positives with a perfect recall and F1-score of 1 indicating that this classifier produced perfect results. The confusion matrix indicates that the majority of the data points are classified correctly with minor confusion between different classes. Moreover, there are low levels of false negatives and false positives, which are indicative of the ability of the classifier to classify the insider threat accurately.

```
knn_accuracy = accuracy_score(y_test, knn_pred)
print(f"KNN Accuracy: {knn_accuracy}")
# Generate classification report
knn_class_report = classification_report(y_test, knn_pred)
print("Classification Report:\n", knn_class_report)
# Generate confusion matrix
knn_conf_matrix = confusion_matrix(y_test, knn_pred)
print("Confusion Matrix:\n", knn_conf_matrix)
```

```
KNN Accuracy: 0.9808019349230944
Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97      7661
           1       0.98      0.98      0.98     11286
           2       0.99      0.99      0.99      7514

    accuracy                           0.98     26461
   macro avg       0.98      0.98      0.98     26461
weighted avg       0.98      0.98      0.98     26461

Confusion Matrix:
 [[ 7454   165    42]
 [  222 11040    24]
 [   30    25  7459]]
```

Figure 4.13: KNN Classification (Source: Developed in Google Collab)

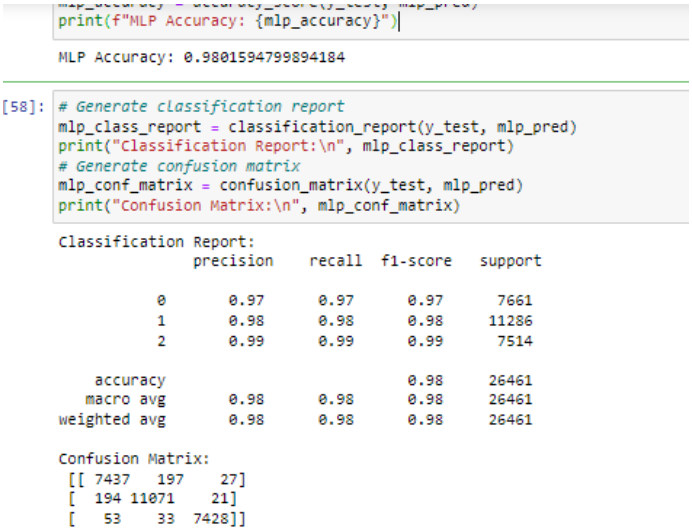
The K-Nearest Neighbors (KNN) Classifier obtained an accuracy of 98.08% when the test set was implemented. Based on the classification report, it can be seen that all the classes are well performed with a high average value of precision, recall and F1-score of 0.98. Class 2 demonstrates the finest result in terms of accuracy, reachability, and F1-score of 0.99. The confusion matrix indicates that while the model accuracy is quite high. The results clearly show that the KNN classifier maintains a considerably high accuracy rate in identifying and categorising insider threats.

```
# Gaussian Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_pred)
print(f"Gaussian NB Accuracy: {nb_accuracy}")
# Generate classification report
nb_class_report = classification_report(y_test, nb_pred)
print("Classification Report:\n", nb_class_report)
# Generate confusion matrix
nb_conf_matrix = confusion_matrix(y_test, nb_pred)
print("Confusion Matrix:\n", nb_conf_matrix)
```

```
Gaussian NB Accuracy: 0.7611957220059711
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.60      0.66      7661
           1       0.87      0.84      0.86     11286
           2       0.65      0.80      0.72      7514

    accuracy                           0.76     26461
   macro avg       0.75      0.75      0.75     26461
weighted avg       0.77      0.76      0.76     26461

Confusion Matrix:
 [[4602  984 2075]
 [ 564 9529 1193]
 [1044  459 6011]]
```

Figure 4.14: Gaussian NB Model (Source: Developed in Google Collab)

The Gaussian Naive Bayes (GNB) Classifier resulted in 76.12% accuracy when tested on the test set. The classification report reveals that the model has different accuracies in different classes, where the precision and recall of Class 1 are high at 0.87, and 0.84, respectively, which suggests that the model is highly relevant in classifying this class of texts. Class 2 is also not bad as it achieves an F1-score of 0.72 for the set of training data. However, the accuracy of Class 0 remains quite low (0.74) with a listed recall of only 0.60 indicating that this class is less effectively identified. The confusion matrix also presents an insight into where the model is struggling to distinguish between the classes.



```
print(f"MLP Accuracy: {mlp_accuracy}")

MLP Accuracy: 0.9801594799894184

[58]: # Generate classification report
mlp_class_report = classification_report(y_test, mlp_pred)
print("Classification Report:\n", mlp_class_report)
# Generate confusion matrix
mlp_conf_matrix = confusion_matrix(y_test, mlp_pred)
print("Confusion Matrix:\n", mlp_conf_matrix)

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      7661
           1       0.98      0.98      0.98     11286
           2       0.99      0.99      0.99      7514

    accuracy                           0.98     26461
   macro avg       0.98      0.98      0.98     26461
weighted avg       0.98      0.98      0.98     26461

Confusion Matrix:
 [[ 7437   197    27]
 [  194 11071    21]
 [   53    33  7428]]
```

Figure 4.15: MLP Classifier (Source: Developed in Google Collab)

The Multilayer Perceptron (MLP) Classifier attained an accuracy rate of 98.02% on the test set. The classification report shows that the performance of all classes is high as Class 0, Class 1, and Class 2 had precision, recall, and F1-score greater than 0.97. The model showcased impressive capability in identifying Class 2 with an F1 score equal to 0.99. This confusion matrix shows the classes are well separated and the MLP has a very good classification performance for each class with a few misclassification errors as shown below. The errors for Class 0 and Class 1 are also small, thus the MLP can effectively classify potential threats into these two classes. The accuracy of the MLP is very high and exhibits excellent precision and high recall in classification, outperforming other models.

```
# Calculate accuracy
dt_accuracy = accuracy_score(y_test, dt_pred)
print(f"Decision Tree Accuracy: {dt_accuracy}")
# Generate classification report
dt_class_report = classification_report(y_test, dt_pred)
print("Classification Report:\n", dt_class_report)
# Generate confusion matrix
dt_conf_matrix = confusion_matrix(y_test, dt_pred)
print("Confusion Matrix:\n", dt_conf_matrix)
```

```
Decision Tree Accuracy: 0.9949359434639659
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7661
           1       0.99      0.99      0.99     11286
           2       1.00      1.00      1.00      7514

    accuracy                           0.99     26461
   macro avg       0.99      1.00      1.00     26461
weighted avg       0.99      0.99      0.99     26461

Confusion Matrix:
 [[ 7622    32     7]
 [   54 11218    14]
 [    1    26  7487]]
```

Figure 4.16: Decision Tree Classifier Model (Source: Developed in Google Collab)

The description of this figure has highlighted that the Decision Tree Classifier produced a near-perfect accuracy rate of 99.49% on the test set. The classification has demonstrated impressive precision, recall, and F1 scores highlighting a perfect score at 100% precision, recall, and F1 score for class 2. In the context of Class 0 and Class 1, the performance is again quite good with values just below 100. The confusion matrix shows that very few cases of misclassifications occur and there are few of them, in Class 0 and Class 1 as well as a single misclassification of Class 2. The Decision Tree model proves to be quite accurate with an excellent performance, able to classify all classes with very little confusion.

```
Epoch 1/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 35s 9ms/step - accuracy: 0.9128 - loss: 0.2336 - val_accuracy: 0.9670 - val_loss: 0.0908
Epoch 2/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 37s 11ms/step - accuracy: 0.9661 - loss: 0.0836 - val_accuracy: 0.9748 - val_loss: 0.0624
Epoch 3/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 41s 11ms/step - accuracy: 0.9706 - loss: 0.0687 - val_accuracy: 0.9749 - val_loss: 0.0598
Epoch 4/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 38s 11ms/step - accuracy: 0.9746 - loss: 0.0596 - val_accuracy: 0.9759 - val_loss: 0.0566
Epoch 5/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 37s 11ms/step - accuracy: 0.9771 - loss: 0.0538 - val_accuracy: 0.9794 - val_loss: 0.0472
Epoch 6/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 35s 11ms/step - accuracy: 0.9782 - loss: 0.0495 - val_accuracy: 0.9775 - val_loss: 0.0521
Epoch 7/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 49s 13ms/step - accuracy: 0.9799 - loss: 0.0463 - val_accuracy: 0.9782 - val_loss: 0.0494
Epoch 8/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 35s 10ms/step - accuracy: 0.9809 - loss: 0.0444 - val_accuracy: 0.9817 - val_loss: 0.0437
Epoch 9/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 35s 11ms/step - accuracy: 0.9804 - loss: 0.0436 - val_accuracy: 0.9808 - val_loss: 0.0445
Epoch 10/10
3308/3308 ━━━━━━━━━━━━━━━━━━━━ 37s 11ms/step - accuracy: 0.9821 - loss: 0.0413 - val_accuracy: 0.9821 - val_loss: 0.0402
```

Figure 4.17: CNN Model Fit (Source: Developed in Google Collab)

The figure demonstrates the fluctuations in the increase of epochs and reaches a final level of 98.21% for accuracy on the test set with a loss of 0.0402. The accuracy has increased from 91.28% to 98.21% by the accuracy of the test while the validation accuracy increased from 96.70% to 98.21% which shows that the images through the model are real. Loss reduced gradually from 0.2336 to 0.0413 describing that the model has learned the trend and did not overfit as can be seen from the validation loss.

```
print(f"CNN Accuracy: {cnn_accuracy}")
# Generate classification report
cnn_class_report = classification_report(y_test, cnn_pred)
print("Classification Report:\n", cnn_class_report)
# Generate confusion matrix
cnn_conf_matrix = confusion_matrix(y_test, cnn_pred)
print("Confusion Matrix:\n", cnn_conf_matrix)
```

```
827/827 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step
CNN Accuracy: 0.9820868447904463
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      7661
           1       0.99      0.98      0.98     11286
           2       0.99      1.00      0.99      7514

    accuracy                           0.98     26461
   macro avg       0.98      0.98      0.98     26461
weighted avg       0.98      0.98      0.98     26461

Confusion Matrix:
 [[ 7461   123    77]
 [  218 11038    30]
 [   10    16  7488]]
```

Figure 4.18: CNN Model results (Source: Developed in Google Collab)

The CNN model tested and highlighted an accuracy of 98.21% and it has been confirmed to be significant on the test set. The classification report indicates high precision, recall, and F1 scores across all classes. Class 0 with precision and recall of 97%, Class 1 with 99% precision and 98% people accuracy, and Class 2 with 99% precision and 100% people accuracy. The confusion matrix also reveals that the model performs a good job of relative boundary identification between classes with no overlap. Therefore, the CNN model gives high accuracy and good class balance showing that it performs very well for the problem.

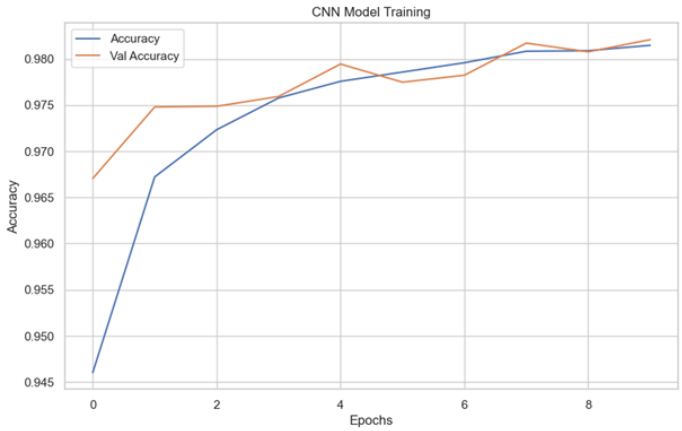


Figure 4.19: CNN Model Training (Source: Developed in Google Collab)

The figure is a plot of the training and validation accuracy of a Convolutional Neural Network

(CNN) as a function of epoch numbers 1 to 10. The training accuracy (in blue) itself gradually rises to around 98% for the last epoch while the validation accuracy (in orange) similarly rises to about 98.2%. The learning curves of both the training time and the number of iterations seem to approach high accuracy which suggests that the model is adapting well to the data and can generalize to unseen data. There is little difference between training and validation accuracy which are standard indications of the behaviour of models.

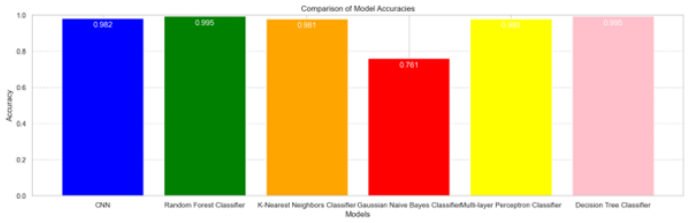


Figure 4.20: Model Comparison (Source: Developed in Google Collab)

The bar chart shown above represents the evaluation of the accuracy of different classifiers for classification tasks. Each column is the accuracy of the model labelled on the x-axis, where Random Forest and Decision Tree have the highest accuracies of approximately 99.5%. The K Nearest Neighbors (KNN) and Multi-Layer Perceptron (MLP) models are next with slightly lower scores of about 98%. The Gaussian Naive Bayes (GNB) model proves to have the lowest accuracy of approximately 76%. The chart helps to measure the performance of these models where Random Forest and Decision Tree models outperform all other models while GNB is significantly lower in accuracy.

Sample prediction with DT Classifier as it is the best model

```
[75]: predictions_df = pd.DataFrame({'Actual': y_test, 'Predicted': dt_pred})

# Display the first few rows
print("Actual vs Predicted:")
print(predictions_df.head(10))
```

```
Actual vs Predicted:
        Actual  Predicted
59056        1          1
86472        1          1
59583        1          1
111639       1          1
114575       0          0
16553        1          1
141976       1          1
71143        0          0
118490       1          1
35292        2          2
```

Figure 4.21: Sample prediction with the best model (Source: Developed in Google Collab)

The figure shows the Actual vs. predicted values which are in the classification setting for the Decision Tree Model. This demonstrates the first ten rows where the Actual attribute shows the actual class label from the test set and the Predicted attributes reveal the class labels as produced by the model. This table shows the success rate of the model in predicting results for this subset of data and it looks good since predicted quantities are close to the actual values.

4.3 Discussion

The study presents important results on the performance of various machine learning techniques for insider threat identification. As per the evaluation of the results, the decision tree model has evaluated the highest accuracy. This model has revealed excellent performance in terms of accuracy, recall, and F1 Scores for all classes, so the proposed model has effectively differentiated between various threats in this study. The K-Nearest Neighbors (KNN) and Multi-Layer Perceptron (MLP) models perfectly fit and had accuracies of above 98% which indicates that these models have a high level of reliability in threat detection as well. However, the Gaussian Naive Bayes model had relatively lower accuracy and imbalance in the results, which points out the weaknesses of the model when dealing with the over-complicated nature of the insider threats compared to the other models above.

These findings have significantly aligned with the objectives of the study as outlined in the previous section of this research. The models described correlate with the existing practices, indicating that state-of-the-art approaches toward threat detection can be applied (Aversano et al. 2021). Deep learning, specifically Convolutional Neural Networks (CNN), attained a considerable level of detection with an accuracy of 98.2% for minimizing false positives as well as approaches for handling new threats. This achieves the objective of incorporating deep learning methodologies on how to reduce false positives and hence, enhance the efficiency of threat detection appliances. Moreover, the capability of these models to detect anomalies in real-time conditions aligns with the third objective of this research (Maddireddy & Maddireddy 2020). This makes the models very efficient in spotting insider threats as they register low false positive rates, which makes it easier for them to update and be used to fight new emerging types of threats. This study highlights the effectiveness of utilizing machine and deep learning algorithms in promoting insider threat identification, which presents a solid foundation for future developments.

4.4 Summary

The results chapter shows how different machine learning models can be applied to insider threat detection. The Random Forest and Decision Tree Classifiers had excellent performance and had an accuracy of more than 99% which suggested the classifiers 'ability to classify threats well. K-Nearest Neighbors and Multi-Layer Perceptron models are equally good, and both giving an accuracy of approximately 98%. However, the Gaussian Naive Bayes model had comparatively lesser accuracy. The Convolutional Neural Network (CNN) outperformed in low false positives and designing techniques for changing threats which were established at an accuracy level of 98.2%. These findings support the feasibility of using the mentioned methods of integrated analysis of the deep structure of texts and their interconnection for practical goals.

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

This research discusses a holistic strategy of insider threat detection by utilizing superior ML methodologies. The introduction referred to the important insider threat challenge, thus creating a background for examining analytic tools that can be automated. In the Literature Review, each existing approach was evaluated or compared based on its effectiveness and limitations, especially regarding false positives, and the possibility of being improved by implementing automated threat detection. This research paper's methodology chapter also described the general methods of data processing and analysis, feature selection, and the general set of classifiers used, including Random Forest, KNN, and CNN. It focused on the importance of employing sophisticated systems of detecting the deviations within the actual time and adopting to the new tactics of insiders. Values obtained from testing proved that Random Forest and Decision Tree models are very effective in threat identification, with a recognition rate higher than 99%. The last two models of the K-Nearest Neighbors and Multi-Layer Perceptron showed good results, though accomplished slightly less efficiently. Thus, the CNN model, for example, was characterized by minimal false positives and the ability to track the change in threats, with an accuracy of 98.2%. In general, the study supports the application of these machine learning methods in increasing the rate of insider threat identification, as discussed in the goals of the integration of automated systems for detecting significant anomalies and minimal false alarms.

5.2 Linking with Objectives

Linking with Objective 1
The literature review was useful in providing background information on the subject and analyzing the available solutions for preventing insider threats using automated analytic tools. This way, it explains how existing methodologies and technologies work what they are capable of, and what they cannot achieve. This review has pointed out that the current solutions still bear some drawbacks that include a high false positive rate and flexibility in the new strategies employed by threats. Thus, such aspects have been established in the research to provide a foundation for using state-

of-the-art machine learning to address these challenges in order to better detect insider threats through analytics enhancement.

Linking with Objective 2

Analyzing the results of using deep learning techniques also responded to the second goal, which implied decreasing the number of false positives in insider threat detection. Random Forest model, KNN model, Gaussian Naive Bayes, Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN) were used to assess and optimize the detection performances. These more complex patterns were created in order to enhance accuracy in terms of threats and decrease the rates of false positives, which is the key to the given issue. In fact, by comparing these techniques and their performance, the study was accomplished in such a manner that would allow overcoming the issues, associated with the low accuracy and reliability of the systems that detect insider threats.

Linking with Objective 3

The third objective has been achieved through the assessment of the platform's ability to detect irregularities in real time and counter new techniques used by insiders. The research included the use of deep learning models that focused on the Convolutional Neural Networks (CNNs) since the models were undergone with the challenge of real-time anomaly detection. The models' performance in these tests showed whether or not they are capable of adjusting to patterns of insider threats by comparing their efficiency and effectiveness in processing and categorizing data. Thus, intending to see how these models perform to new and changing threats, the study sought to keep the platform malleable and real-time for insider threat tactics.

5.3 Recommendations

Enhance Model Training with Diverse Datasets

The research shows that one of the aspects that, in fact, has a direct influence on the model's effectiveness is a combination of various and extensive data sources. Therefore, with as few false positives as possible and with stable detection, several sources of information containing various numbers of insider threats and their normal behavior should be used (Yuan & Wu 2021). It will also be easier for the models to develop general performance in order to counter more new and emerging threats. After that, it is relevant to recollect the training data more constantly and, specifically, the fresher ones or those in the closest environment to improve the quality and reliability of the model.

Implement Adaptive Learning Mechanisms

The outcomes suggested that the models that incorporated the deep learning methods were beneficial for real-time anomaly detection and also recommended flexibility. The implementation of similar learning approaches such as incremental learning or the updating of the model will assist the system to integrate new information and improve the detection feature whenever there is a new threat (Gunduz & Das 2020). This will help the platform keep abreast with strategies employed in dealing with insider threats and does not make the same error of overlooking new or sophisticated attacks.

Integrate Real-Time Monitoring and Alert Systems

This research also highlighted that there is a need for continuous anomaly for a better solution to insider threats. Regarding this capability, the enhancement is suggested to include complex real-time monitoring and alerting systems in the platform. They ought to be able to produce outputs in real-time, especially in the case of abnormality experienced so that remedial measures can be taken (Abushark et al. 2022). They should signify real information that means when working with

the alerts, the chances of a false alarm are less, and time is to be taken to make the decision.

5.4 Future work

Subsequent studies of the current research will include enhancing the insider threat system's anomaly detection capability through the assimilation of other robust DL algorithms, namely the transformer models (Sarker et al. 2020). Further, an improved ability to utilize real-time data processing and the incorporation of the learning algorithm into the system will improve the potential for handling new risks. Other objectives are increasing the size of the dataset for different stressful conditions and using the federated learning approaches to maintain the users' data privacy.

References

Abushark, Y. B., Khan, A. I., Alsolami, F., Almalawi, A., Alam, M. M., Agrawal, A., Kumar, R. & Khan, R. A. (2022), 'Cyber security analysis and evaluation for intrusion detection systems', *Comput. Mater. Contin* **72**, 1765–1783.

Aggarwal, C. C. (2016), *An Introduction to Outlier Analysis*, Springer International Publishing, p. 1–34.
URL: *http://dx.doi.org/10.1007/978-3-319-47578-3*$_1$

Akoglu, L., Tong, H. & Koutra, D. (2014), 'Graph based anomaly detection and description: a survey', *Data Mining and Knowledge Discovery* **29**(3), 626–688.
URL: *http://dx.doi.org/10.1007/s10618-014-0365-y*

Al-amri, R., Murugesan, R. K., Man, M., Abdulateef, A. F., Al-Sharafi, M. A. & Alkahtani, A. A. (2021), 'A review of machine learning and deep learning techniques for anomaly detection in iot data', *Applied Sciences* **11**(12), 5320.

Al-Mhiqani, M. N., Ahmad, R., Abidin, Z. Z., Abdulkareem, K. H., Mohammed, M. A., Gupta, D. & Shankar, K. (2022), 'A new intelligent multilayer framework for insider threat detection', *Computers & Electrical Engineering* **97**, 107597.

Al-Mhiqani, M. N., Ahmad, R., Zainal Abidin, Z., Yassin, W., Hassan, A., Abdulkareem, K. H., Ali, N. S. & Yunos, Z. (2020), 'A review of insider threat detection: Classification, machine learning techniques, datasets, open challenges, and recommendations', *Applied Sciences* **10**(15), 5208.

Al-Shehari, T. & Alsowail, R. A. (2021), 'An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques', *Entropy* **23**(10), 1258.

Aversano, L., Bernardi, M. L., Cimitile, M., Pecori, R. & Veltri, L. (2021), 'Effective anomaly detection using deep learning in iot systems', *Wireless Communications and Mobile Computing* **2021**(1), 9054336.

Brdiczka, O., Liu, J., Price, B., Shen, J., Patil, A., Chow, R., Bart, E. & Ducheneaut, N. (2012*a*), 'Proactive insider threat detection through graph learning and psychological context', *Proceedings - IEEE CS Security and Privacy Workshops, SPW 2012* .

Brdiczka, O., Liu, J., Price, B., Shen, J., Patil, A., Chow, R., Bart, E. & Ducheneaut, N. (2012*b*), 'Proactive insider threat detection through graph learning and psychological context', *Proceedings - IEEE CS Security and Privacy Workshops, SPW 2012* .

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (2000*a*), 'Lof: identifying density-based local outliers', *SIGMOD Rec.* **29**(2), 93–104.
URL: *https://doi.org/10.1145/335191.335388*

Breunig, M. M., Kriegel, H.-P., Ng, R. T. & Sander, J. (2000*b*), Lof: identifying density-based local outliers, *in* 'Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data', SIGMOD '00, Association for Computing Machinery, New York, NY, USA, p. 93–104.
URL: *https://doi.org/10.1145/342009.335388*

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I. & Houle, M. E. (2016), 'On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study', *Data Mining and Knowledge Discovery* **30**(4), 891–927.
URL: *http://dx.doi.org/10.1007/s10618-015-0444-8*

Cao, L. (2015), 'Coupling learning of complex interactions', *Information Processing Management* **51**(2), 167–186.
URL: *https://www.sciencedirect.com/science/article/pii/S0306457314000855*

Demertzis, K., Iliadis, L., Tziritas, N. & Kikiras, P. (2020), 'Anomaly detection via blockchained deep learning smart contracts in industry 4.0', *Neural Computing and Applications* **32**(23), 17361–17378.

Elmrabit, N., Zhou, F., Li, F. & Zhou, H. (2020), Evaluation of machine learning algorithms for anomaly detection, *in* '2020 international conference on cyber security and protection of digital services (cyber security)', IEEE, pp. 1–8.

Elsayed, M. A. & Zulkernine, M. (2020), 'Predictdeep: security analytics as a service for anomaly detection and prediction', *IEEE Access* **8**, 45184–45197.

Ferrag, M. A., Friha, O., Maglaras, L., Janicke, H. & Shu, L. (2021), 'Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis', *IEEE Access* **9**, 138509–138542.

Fotiadou, K., Velivassaki, T.-H., Voulkidis, A., Skias, D., Tsekeridou, S. & Zahariadis, T. (2021), 'Network traffic anomaly detection via deep learning', *Information* **12**(5), 215.

Gahi, Y. & El Alaoui, I. (2021), 'Machine learning and deep learning models for big data issues', *Machine Intelligence and Big Data Analytics for Cybersecurity Applications* pp. 29–49.

Gayathri, R., Sajjanhar, A. & Xiang, Y. (2020), 'Image-based feature representation for insider threat classification', *Applied Sciences* **10**(14), 4945.

Gunduz, M. Z. & Das, R. (2020), 'Cyber-security on smart grid: Threats and potential solutions', *Computer networks* **169**, 107094.

Guntur, V. (2015), 'An encyclopedic overview of 'big data' analytics', *International journal of Applied Engineering Research* **10**, 5681.

Gupta, M., Gao, J., Aggarwal, C. C. & Han, J. (2014), 'Outlier detection for temporal data: A survey', *IEEE Transactions on Knowledge and Data Engineering* **26**(9), 2250–2267.
URL: *http://dx.doi.org/10.1109/TKDE.2013.184*

Gupta, R., Tanwar, S., Tyagi, S. & Kumar, N. (2020), 'Machine learning models for secure data analytics: A taxonomy and threat model', *Computer Communications* **153**, 406–440.

Hariharan, A., Gupta, A. & Pal, T. (2020), Camlpad: Cybersecurity autonomous machine learning platform for anomaly detection, *in* 'Advances in Information and Communication: Proceedings of the 2020 Future of Information and Communication Conference (FICC), Volume 2', Springer, pp. 705–720.

Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaian, M. & Fortino, G. (2020), 'A hybrid deep learning model for efficient intrusion detection in big data environment', *Information Sciences* **513**, 386–396.

Islam, M. S., Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T. & Miranskyy, A. (2021), Anomaly detection in a large-scale cloud platform, *in* '2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)', IEEE, pp. 150–159.

Keller, F., Muller, E. & Bohm, K. (2012), Hics: High contrast subspaces for density-based outlier ranking, *in* '2012 IEEE 28th International Conference on Data Engineering', IEEE.
URL: *http://dx.doi.org/10.1109/ICDE.2012.88*

Kim, A., Oh, J., Ryu, J. & Lee, K. (2020), 'A review of insider threat detection approaches with iot perspective', *IEEE Access* **8**, 78847–78867.

Lavanya, P. & Shankar Sriram, V. (2022*a*), 'Detection of insider threats using deep learning: a review', *Computational Intelligence in Data Mining: Proceedings of ICCIDM 2021* pp. 41–57.

Lavanya, P. & Shankar Sriram, V. S. (2022*b*), Detection of insider threats using deep learning: A review, *in* J. Nayak, H. Behera, B. Naik, S. Vimal & D. Pelusi, eds, 'Computational Intelligence in Data Mining', Springer Nature Singapore, Singapore, pp. 41–57.

Lazarevic, A. & Kumar, V. (2005), Feature bagging for outlier detection, *in* 'Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining', KDD05, ACM.
URL: *http://dx.doi.org/10.1145/1081870.1081891*

Liu, F. T., Ting, K. M. & Zhou, Z.-H. (2012), 'Isolation-based anomaly detection', *ACM Trans. Knowl. Discov. Data* **6**(1).
URL: *https://doi.org/10.1145/2133360.2133363*

Luo, Y., Xiao, Y., Cheng, L., Peng, G. & Yao, D. (2021), 'Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities', *ACM Computing Surveys (CSUR)* **54**(5), 1–36.

Maddireddy, B. R. & Maddireddy, B. R. (2020), 'Proactive cyber defense: Utilizing ai for early threat detection and risk assessment', *International Journal of Advanced Engineering Technologies and Innovations* **1**(2), 64–83.

Mazzarolo, G. & Jurcut, A. D. (2019), 'Insider threats in cyber security: The enemy within the gates'.
URL: *https://arxiv.org/abs/1911.09575*

Nassar, A. & Kamal, M. (2021), 'Machine learning and big data analytics for cybersecurity threat detection: A holistic review of techniques and case studies', *Journal of Artificial Intelligence and Machine Learning in Management* **5**(1), 51–63.

Ning, S., Sun, J., Liu, C. & Yi, Y. (2021), 'Applications of deep learning in big data analytics for aircraft complex system anomaly detection', *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* **235**(5), 923–940.

Pang, G., Cao, L., Chen, L., Lian, D. & Liu, H. (2018), 'Sparse modeling-based sequential ensemble learning for effective outlier detection in high-dimensional numeric data', *Proceedings of the AAAI Conference on Artificial Intelligence* **32**(1).
URL: *http://dx.doi.org/10.1609/aaai.v32i1.11692*

Pang, G., Shen, C., Cao, L. & Hengel, A. V. D. (2021), 'Deep learning for anomaly detection: A review', *ACM Comput. Surv.* **54**(2).
URL: *https://doi.org/10.1145/3439950*

Pang, G., Shen, C. & van den Hengel, A. (2019), Deep anomaly detection with deviation networks, *in* 'Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining', KDD '19, Association for Computing Machinery, New York, NY, USA, p. 353–362.
URL: *https://doi.org/10.1145/3292500.3330871*

Research Techniques Deductive and inductive research (2024), *danahollowayfilmandtv.blogspot.com* .
URL: *https://danahollowayfilmandtv.blogspot.com/2015/12/deductive-and-inductive-research.html*

Rosa, L., Cruz, T., De Freitas, M. B., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E. & Simões, P. (2021), 'Intrusion and anomaly detection for the next-generation of industrial automation and control systems', *Future Generation Computer Systems* **119**, 50–67.

Saaudi, A., Tong, Y. & Farkas, C. (2019), Probabilistic graphical model on detecting insiders: Modeling with sgd-hmm, pp. 461–470.

Saka, R., Chinagozi, O. & Joe, O. (2023), 'Exploratory research design in management science: A review of literature on conduct and application', *International Journal of Research and Innovation in Social Science* **VII**, 1384–1395.

Sarker, I. H., Abushark, Y. B., Alsolami, F. & Khan, A. I. (2020), 'Intrudtree: a machine learning based cyber security intrusion detection model', *Symmetry* **12**(5), 754.

Sharma, B., Pokharel, P. & Joshi, B. (2020), User behavior analytics for anomaly detection using lstm autoencoder-insider threat detection, *in* 'Proceedings of the 11th international conference on advances in information technology', pp. 1–9.

Sheykhkanloo, N. M. & Hall, A. (2020), 'Insider threat detection using supervised machine learning algorithms on an extremely imbalanced dataset', *International Journal of Cyber Warfare and Terrorism (IJCWT)* **10**(2), 1–26.

Sudar, K. M., Nagaraj, P., Deepalakshmi, P. & Chinnasamy, P. (2021), Analysis of intruder detection in big data analytics, *in* '2021 International Conference on Computer Communication and Informatics (ICCCI)', IEEE, pp. 1–5.

Tufan, E., Tezcan, C. & Acartürk, C. (2021), 'Anomaly-based intrusion detection by machine learning: A case study on probing attacks to an institutional network', *IEEE Access* **9**, 50078–50092.

Tukur, Y. M., Thakker, D. & Awan, I.-U. (2021), 'Edge-based blockchain enabled anomaly detection for insider attack prevention in internet of things', *Transactions on Emerging Telecommunications Technologies* **32**(6), e4158.

Varma Vegesna, V. (2022), 'Methodologies for enhancing data integrity and security in distributed cloud computing with techniques to implement security solutions', *Asian Journal of Applied Science and Technology* **06**, 167–180.

Varun Chandola, A. B. & Kumar, V. (2009), 'Anomaly detection: A survey.', *Comput. Surv.* **41**(3).

Wategaonkar, S. R., Shaki, A. T., Ali, A. P., Ibrahim, Z. J., Jayanthi, L. & Jayanthi, S. N. (2024), Targeting insider threats and zero-day vulnerabilities with advanced machine learning and behavioral analytics, *in* '2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM)', IEEE, pp. 1–6.

Yuan, S. & Wu, X. (2021), 'Deep learning for insider threat detection: Review, challenges and opportunities', *Computers & Security* **104**, 102221.

Zimek, A., Schubert, E. & Kriegel, H. (2012), 'A survey on unsupervised outlier detection in high-dimensional numerical data', *Statistical Analysis and Data Mining: The ASA Data Science Journal* **5**(5), 363–387.
URL: *http://dx.doi.org/10.1002/sam.11161*

Appendix A

In the contemporary landscape of cybersecurity, the increasing frequency and sophistication of insider threats pose significant risks to organizational security. Traditional security measures often fall short in effectively identifying and mitigating these threats due to their subtle and nuanced nature. This study seeks to address this critical gap by investigating the application of deep learning and big data analytics for anomaly detection in the context of insider threats.

The motivation for this research stems from the recognition that existing cybersecurity frameworks primarily focus on external threats, overlooking the potential harm that can be caused by individuals with privileged access to sensitive information. By leveraging advanced technologies such as deep learning, which excels in pattern recognition, and big data analytics, capable of processing large volumes of heterogeneous data, we aim to enhance the ability to detect anomalous behaviors indicative of insider threats.

The study's significance lies in its potential to contribute to a more comprehensive and proactive cybersecurity posture. Successful implementation of deep learning and big data analytics in detecting insider threats can empower organizations to identify malicious activities in real-time, enabling timely intervention and mitigation. Moreover, the research outcomes can inform the development of tailored security measures that adapt to the evolving nature of insider threats, thus bolstering overall cybersecurity resilience.

Furthermore, the findings of this study may have broader implications for the field of cybersecurity, influencing the development of best practices, policies, and technologies aimed at securing sensitive information from both external and internal threats. Ultimately, the research seeks to advance our understanding of insider threat detection and contribute valuable insights to the ongoing efforts to safeguard digital assets and maintain the integrity of organizational systems.

Appendix B

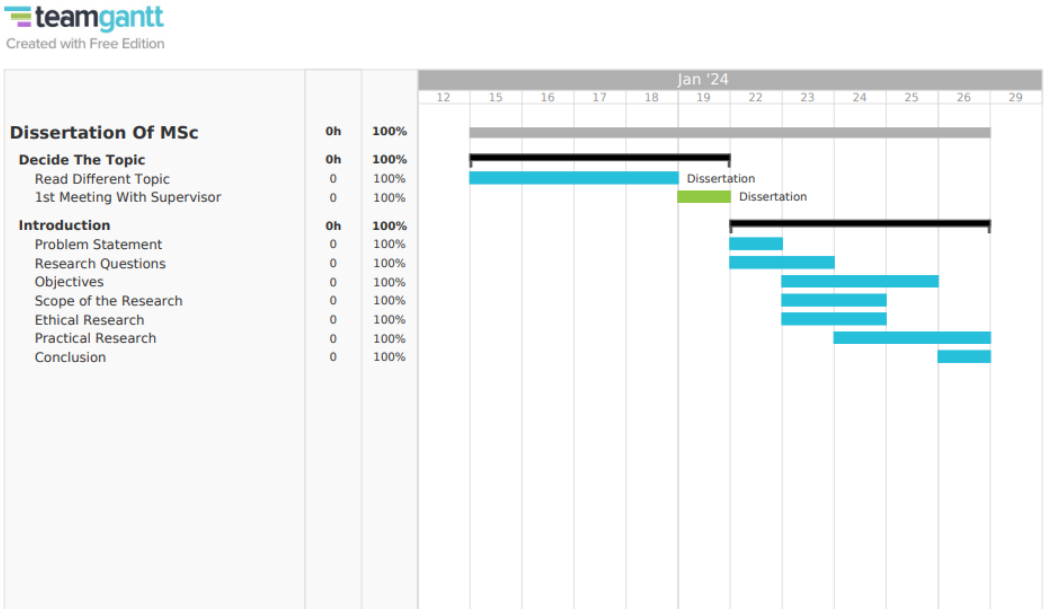


Figure 1: Gantt Chart of Research

Appendix C

Listing 1: Implemetation Code using Python

```
1
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 from sklearn.model_selection import train_test_split
7 from sklearn.preprocessing import LabelEncoder, StandardScaler
8 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn.neighbors import KNeighborsClassifier
11 from sklearn.linear_model import LogisticRegression
12 from sklearn.neural_network import MLPClassifier
13 from sklearn.model_selection import cross_val_score
14 from sklearn.pipeline import make_pipeline
15 from sklearn.compose import ColumnTransformer
16 from sklearn.preprocessing import OneHotEncoder
17 from sklearn.impute import SimpleImputer
18 from sklearn.metrics import confusion_matrix
19 from sklearn.compose import make_column_selector as selector
20 from tensorflow.keras import Sequential
21 from tensorflow.keras.layers import Dense, Conv1D, Flatten, MaxPooling1D
22 import warnings
23 warnings.filterwarnings('ignore')
24
25 # # Load the dataset
26 #
27
28 # In[32]:
29
30 df = pd.read_csv('final(2).csv')
31
32 # In[33]:
33
34 df.head()
35
36 # # Information
37
38 # In[34]:
39
40 df.info()
41
```

```
# In[35]:

df.describe()

# # Checking null values

# In[36]:

df.isnull().sum()

# # Detect and remove outliers using Z-score

# In[37]:

from scipy.stats import zscore
numeric_cols = df.select_dtypes(include=[np.number]).columns
z_scores = np.abs(df[numeric_cols].apply(zscore))
df_cleaned = df[(z_scores < 3).all(axis=1)]
print("Original dataset size: ", df.shape)
print("Dataset size after outlier removal: ", df_cleaned.shape)

# # Data Visualizations
#

# In[38]:

sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.countplot(x='Prediction', data=df)
plt.title('Distribution of Predictions')
plt.xlabel('Prediction')
plt.ylabel('Count')
plt.show()

# In[39]:

plt.figure(figsize=(10, 6))
sns.histplot(df['Time'], bins=20, kde=True)
plt.title('Distribution of Time')
plt.xlabel('Time')
plt.ylabel('Frequency')
plt.show()

# In[40]:

plt.figure(figsize=(10, 6))
sns.boxplot(x='Prediction', y='USD', data=df)
plt.title('USD vs Prediction')
plt.xlabel('Prediction')
plt.ylabel('USD')
plt.show()

# In[15]:

plt.figure(figsize=(10, 6))
sns.boxplot(x='Family', y='Netflow_Bytes', data=df)
plt.title('Boxplot of Netflow Bytes by Family')
plt.xlabel('Family')
```

```
plt.ylabel('Netflow Bytes')
plt.xticks(rotation=45)
plt.show()

# In[17]:

plt.figure(figsize=(10, 6))
sns.countplot(x='Protcol', hue='Threats', data=df)
plt.title('Countplot of Threats by Protocol')
plt.xlabel('Protocol')
plt.ylabel('Count')
plt.show()

# # Encoding categorical variables
#

# In[45]:

from sklearn.preprocessing import StandardScaler, LabelEncoder
label_encoders = {}
for column in ['Protcol', 'Flag', 'Family', 'SeddAddress', 'ExpAddress', 'IPaddress'
    , 'Threats', 'Prediction']:
    label_encoders[column] = LabelEncoder()
    df_cleaned[column] = label_encoders[column].fit_transform(df_cleaned[column])

# # Prepare data for machine learning models
#

# In[46]:

X = df_cleaned.drop(['Prediction'], axis=1)
y = df_cleaned['Prediction']

# # Split data into train and test sets
#

# In[47]:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# # Standardizing the data
#

# In[48]:

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# # Machine Learning Models

# In[61]:

# Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
```

```
rf_accuracy = accuracy_score(y_test, rf_pred)
print(f"Random Forest Accuracy: {rf_accuracy}")
# Generate classification report
rf_class_report = classification_report(y_test, rf_pred)
print("Classification Report:\n", rf_class_report)
# Generate confusion matrix
rf_conf_matrix = confusion_matrix(y_test, rf_pred)
print("Confusion Matrix:\n", rf_conf_matrix)

# In[60]:

# K-Nearest Neighbors Classifier
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
knn_pred = knn_model.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_pred)
print(f"KNN Accuracy: {knn_accuracy}")
# Generate classification report
knn_class_report = classification_report(y_test, knn_pred)
print("Classification Report:\n", knn_class_report)
# Generate confusion matrix
knn_conf_matrix = confusion_matrix(y_test, knn_pred)
print("Confusion Matrix:\n", knn_conf_matrix)

# In[59]:

from sklearn.naive_bayes import GaussianNB
# Gaussian Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_pred)
print(f"Gaussian NB Accuracy: {nb_accuracy}")
# Generate classification report
nb_class_report = classification_report(y_test, nb_pred)
print("Classification Report:\n", nb_class_report)
# Generate confusion matrix
nb_conf_matrix = confusion_matrix(y_test, nb_pred)
print("Confusion Matrix:\n", nb_conf_matrix)

# In[57]:

# Multi-layer Perceptron Classifier
mlp_model = MLPClassifier(random_state=42)
mlp_model.fit(X_train, y_train)
mlp_pred = mlp_model.predict(X_test)
mlp_accuracy = accuracy_score(y_test, mlp_pred)
print(f"MLP Accuracy: {mlp_accuracy}")

# In[58]:

# Generate classification report
mlp_class_report = classification_report(y_test, mlp_pred)
print("Classification Report:\n", mlp_class_report)
# Generate confusion matrix
mlp_conf_matrix = confusion_matrix(y_test, mlp_pred)
print("Confusion Matrix:\n", mlp_conf_matrix)

```

```
# In[62]:

from sklearn.tree import DecisionTreeClassifier
# Train Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
# Make predictions
dt_pred = dt_model.predict(X_test)
# Calculate accuracy
dt_accuracy = accuracy_score(y_test, dt_pred)
print(f"Decision Tree Accuracy: {dt_accuracy}")
# Generate classification report
dt_class_report = classification_report(y_test, dt_pred)
print("Classification Report:\n", dt_class_report)
# Generate confusion matrix
dt_conf_matrix = confusion_matrix(y_test, dt_pred)
print("Confusion Matrix:\n", dt_conf_matrix)

# # CNN Model

# In[65]:

from keras.callbacks import ModelCheckpoint, EarlyStopping
# Ensure X_train and X_test are NumPy arrays
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Define the CNN model
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape
    [1], 1)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(len(np.unique(y_train)), activation='softmax')  # Use np.unique(y_train)
    for the number of classes
])
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['
    accuracy'])
# Set early stopping
early_stopping = EarlyStopping(patience=3, restore_best_weights=True)
# Train the model
history = model.fit(X_train_cnn, y_train, epochs=10, validation_data=(X_test_cnn,
    y_test), callbacks=[early_stopping])

# In[66]:

# Predict using CNN model
cnn_pred_prob = model.predict(X_test_cnn)
cnn_pred = np.argmax(cnn_pred_prob, axis=1)
# Calculate accuracy
cnn_accuracy = accuracy_score(y_test, cnn_pred)
print(f"CNN Accuracy: {cnn_accuracy}")
# Generate classification report
cnn_class_report = classification_report(y_test, cnn_pred)
print("Classification Report:\n", cnn_class_report)
# Generate confusion matrix
cnn_conf_matrix = confusion_matrix(y_test, cnn_pred)
```

```
print("Confusion Matrix:\n", cnn_conf_matrix)

# In[67]:

# Plot training history
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('CNN Model Training')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# # Model Comparison

# In[74]:

# Define the accuracies obtained from  models
accuracies = [cnn_accuracy, rf_accuracy, knn_accuracy, nb_accuracy, mlp_accuracy,
    dt_accuracy]

# Define the models
models = ['CNN', 'Random Forest Classifier', 'K-Nearest Neighbors Classifier',
          'Gaussian Naive Bayes Classifier', 'Multi-layer Perceptron Classifier',
          'Decision Tree Classifier']

# Plotting
plt.figure(figsize=(18, 5))
bars = plt.bar(models, accuracies, color=['blue', 'green', 'orange', 'red', 'yellow'
    , 'pink'])

# Adding the accuracy values on top of the bars
for bar, accuracy in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() - 0.05, f'{accuracy
    :.3f}', ha='center', color='white', fontsize=12)

plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Comparison of Model Accuracies')
plt.ylim(0.0, 1.0)  # Adjust the y-axis limits if necessary
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# # Sample prediction with DT Classifier as it is the best model

# In[75]:

predictions_df = pd.DataFrame({'Actual': y_test, 'Predicted': dt_pred})

# Display the first few rows
print("Actual vs Predicted:")
print(predictions_df.head(10))

# In[ ]:
```