

Network Anomalies Detection Using Traffic Patterns Analysis

D. P. P. De Silva

2024



Network Anomalies Detection Using Traffic Patterns Analysis

**A Thesis Submitted for the Degree of Master of
Computer Science**



D. P. P. De Silva

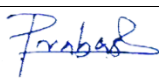
University of Colombo School of Computing

2024

DECLARATION

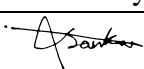
Name of the student: D. P. P. De Silva
Registration number: 2020/MCS/016
Name of the Degree Programme: Master of Computer Science
Project/Thesis title: Network Anomalies Detection Using Traffic Patterns Analysis

1. The project/thesis is my original work and has not been submitted previously for a degree at this or any other University/Institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.
2. I understand what plagiarism is, the various types of plagiarism, how to avoid it, what my resources are, who can help me if I am unsure about a research or plagiarism issue, as well as what the consequences are at University of Colombo School of Computing (UCSC) for plagiarism.
3. I understand that ignorance is not an excuse for plagiarism and that I am responsible for clarifying, asking questions and utilizing all available resources in order to educate myself and prevent myself from plagiarizing.
4. I am also aware of the dangers of using online plagiarism checkers and sites that offer essays for sale. I understand that if I use these resources, I am solely responsible for the consequences of my actions.
5. I assure that any work I submit with my name on it will reflect my own ideas and effort. I will properly cite all material that is not my own.
6. I understand that there is no acceptable excuse for committing plagiarism and that doing so is a violation of the Student Code of Conduct.

Signature of the Student	Date (DD/MM/YYYY)
	25/10/2024

Certified by Supervisor(s)

This is to certify that this project/thesis is based on the work of the above-mentioned student under my/our supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

	Supervisor 1	Supervisor 2
Name	Dr. Asanka P. Sayakkara	
Signature		
Date	2024-10-25	

ACKNOWLEDGEMENTS

I am deeply thankful to my supervisor, Dr. Asanka P. Sayakkara for his invaluable support and guidance throughout my research. His mentorship has been a source of inspiration, fostering an environment of curiosity and resilience. I appreciate his commitment to encouraging independent thinking and dedication to nurturing my research work. Thank you for being more than a mentor – a collaborator and a guiding force.

ABSTRACT

The exponential growth of Internet users over the past decades underscores the pivotal role of the Internet in modern life. However, this surge in Internet usage has led to a corresponding increase in cyber security attacks, especially given the rise of digital currencies. The reliance on signature-based detection and challenges posed by SSL/TLS encryption highlights the pressing need for advanced approaches to network security. Traffic behaviour analysis solutions prove effective in addressing challenges. Models detecting network anomalies through traffic behaviours are crucial in overcoming issues. The methodology involves training machine learning-based models on a comprehensive dataset with diverse traffic features, ensuring accurate anomaly detection.

This research addresses the critical issue of network security by proposing an innovative approach to anomaly detection using advanced machine learning techniques and leveraging a recently collected, up-to-date dataset. The trained model underwent evaluation using the same test dataset across all selected algorithms, ensuring a fair comparison. For the performance evaluation of each algorithm, a comprehensive set of evaluation metrics was employed, including accuracy, precision, recall, F1-score, and the area under the Receiver Operating Characteristic (ROC) curve. Utilizing a labeled dataset encompassing various attack types, the proposed traffic pattern-based anomaly detection experiments achieved remarkable results, with each tested machine learning algorithm surpassing 95% accuracy with binary classes. Random Forest, XGBoost, and K-Nearest Neighbors emerged as the top performers, boasting validation accuracy rates of 99.64%, 99.61%, and 99.05%, respectively. Furthermore, these algorithms performed well even in the presence of infrequent anomaly events. This research significantly advances network anomaly detection, offering valuable insights for cybersecurity practitioners. The study introduces a versatile and adaptable approach to effectively safeguard against dynamic cyber threats in the digital era.

TABLE OF CONTENTS

DECLARATION	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	ix
LIST OF ACRONYMS	x
1. CHAPTER - INTRODUCTION	1
1.1 Problem Background	1
1.2 Motivation	3
1.3 Statement of the problem	4
1.4 Research Aims and Objectives	5
1.4.1 Aim	5
1.4.2 Objectives	5
1.5 Scope	5
1.6 Significance of the Research	6
1.7 Structure of the Thesis	7
2. CHAPTER - LITERATURE REVIEW	8
2.1 Introduction	8
2.2 Theoretical Framework	9
2.2.2 Common Anomalies in Network Security	9
2.2.4 Machine Learning and Anomaly Detection	15
2.3 Related Work	18
2.4 Summary	24
3. CHAPTER - METHODOLOGY	25
3.1 Research Strategy and Ethics	25
3.2 Research Design	26
3.3 Research Evaluation process	28
3.3.1 Evaluation Metrics and Measures	28
3.3.2 Comparison with Existing Solutions	29
3.4 Data Collection	30
3.4.1 Integration of Well-known Datasets	30
4. CHAPTER - EVALUATION AND RESULTS	31
4.1 Training the Model	31

4.2 Testing the Trained Data Model	36
4.3 Evaluation of trained model performances in different ML Algorithms.....	40
4.3.1 Algorithms Selection and Considerations	40
4.3.2 Unsupervised learning algorithms	41
4.3.3 Supervised learning algorithms	47
4.3.4 Machine Learning techniques behaviour analysis with infrequent anomaly events	63
4.3.5 Evaluation Summary of trained model performance using different ML methods	67
5. CHAPTER - CONCLUSION AND FUTURE WORK	72
5.1 Conclusion.....	72
5.2 Limitations	73
5.3 Novelty and implications of the research	75
5.4 Future directions	76
APPENDICES.....	I
Appendix 1 - Trained model programs.....	I
Appendix 2 - Data set sample.....	III
REFERENCES.....	VIII

LIST OF FIGURES

Figure 1.1: Exponential growth of Internet users (statista, 2023)	1
Figure 2.1: Areas in cyber security (ECS, 2020)	10
Figure 2.2: Procedure of session hijacking (Ghosh, A., & Senthilrajan, A., 2020).....	11
Figure 2.3: Cross-site scripting attack (Singh and Jain, 2018).....	13
Figure 3.1: Research design flow chart	27
Figure 3.2: Research evaluation process flow chart	29
Figure 4.1: Program used for the trained model generation.....	31
Figure 4.2: Program used for the trained model performance classification.....	32
Figure 4.3: Dataset Features with descriptions (Neto, E.C. et al., 2023)	33
Figure 4.4: Training model program output with binary classes	34
Figure 4.5: Training model program output with multiple classes	35
Figure 4.6: Test dataset	36
Figure 4.7: Testing program for the experiment data	38
Figure 4.8: Multiclass dataset used trained model test outcome	39
Figure 4.9: Binary class dataset used trained model test outcome	39
Figure 4.10: Isolation forest algorithm based trained model function	41
Figure 4.11: Classification report for the isolation forest algorithm	41
Figure 4.12: Confusion matrix and ROC Curve for isolation forest algorithm	42
Figure 4.13: LOF algorithm based trained model function.....	43
Figure 4.14: Classification report for the LOF algorithm	43
Figure 4.15: Confusion matrix and ROC Curve for LOF algorithm	44
Figure 4.16: Autoencoders algorithm based trained model function	45
Figure 4.17: Classification report for the Autoencoders algorithm.....	45
Figure 4.18: Confusion matrix and ROC Curve for Autoencoders algorithm	46
Figure 4.19: Random forest algorithm based trained model function	47
Figure 4.20: Classification report for the random forest algorithm.....	47
Figure 4.21: Confusion matrix and ROC Curve for random forest algorithm	48
Figure 4.22: Logistic Regression algorithm based trained model function.....	49
Figure 4.23: Classification report for the Logistic Regression algorithm	49
Figure 4.24: Confusion matrix and ROC Curve for Logistic Regression algorithm	50
Figure 4.25: Decision Trees algorithm based trained model function	51
Figure 4.26: Classification report for the Decision Trees algorithm.....	51
Figure 4.27: Confusion matrix and ROC Curve for Decision Trees algorithm.....	52
Figure 4.28: K-Nearest Neighbors algorithm based trained model function.....	53
Figure 4.29: Classification report for the K-Nearest Neighbors algorithm	53
Figure 4.30: Confusion matrix and ROC Curve for K-Nearest Neighbors algorithm.....	54

Figure 4.31: Naive Bayes algorithm based trained model function	55
Figure 4.32: Classification report for the Naive Bayes algorithm	55
Figure 4.33: Confusion matrix and ROC Curve for Naive Bayes algorithm	56
Figure 4.34: AdaBoost algorithm based trained model function	57
Figure 4.35: Classification report for the AdaBoost algorithm.....	57
Figure 4.36: Confusion matrix and ROC Curve for AdaBoost algorithm.....	58
Figure 4.37: XGBoost algorithm based trained model function	59
Figure 4.38: Classification report for the XGBoost algorithm.....	59
Figure 4.39: Confusion matrix and ROC Curve for XGBoost algorithm	60
Figure 4.40: Perceptron algorithm based trained model function	61
Figure 4.41: Classification report for the Perceptron algorithm	61
Figure 4.42: Confusion matrix and ROC Curve for Perceptron algorithm	62
Figure 4.43: Results for the Random forest algorithm with infrequent anomaly events	63
Figure 4.44: Results for the Decision Trees algorithm with infrequent anomaly events.....	64
Figure 4.45: Results for the Isolation Forests algorithm with infrequent anomaly events	65
Figure 4.46: Validation Accuracy of Models based on ML Algorithms	68
Figure 4.47: Evaluation matrices for each ML algorithms	69
Figure 4.48: Validation Accuracy of Models based on ML Algorithms across multiple classes cooperating infrequent anomalies.....	70
Figure 4.49: Evaluation matrices for each ML algorithms across multiple classes cooperating infrequent anomalies	71
Figure A.1: Sample evaluation model program for random forest algorithm	I
Figure A.2: Sample evaluation model program for Isolation forest algorithm	II
Figure A.3: Labeled data set with features and classes part i	III
Figure A.4: Labeled data set with features and classes part ii	IV
Figure A.5: Labeled data set with features and classes part iii	V
Figure A.6: Labeled data set with features and classes part iv.....	VI
Figure A.7: Statistical information about various features in the dataset (Neto, E.C. et al., 2023)	VII

LIST OF TABLES

Table 4.1: Model performance in Random forest algorithm with binary and multiple classes	36
Table 4.2: Evaluation scores summary, green highlights unsupervised methods, purple highlights supervised methods across binary classes.	67
Table 4.3: Evaluation scores summary, with green highlighting for unsupervised methods and purple highlighting for supervised methods across multiple classes cooperating infrequent anomalies.	69

LIST OF ACRONYMS

AUC	-	Area under the Curve
DoS	-	Denial of Service
DPI	-	Deep Packet Inspection
IoT	-	Internet of Things
ML	-	Machine Learning
ROC	-	Receiver Operating Characteristic
SSL	-	Secure Sockets Layer
SVM	-	Support Vector Machine
TLS	-	Transport Layer Security
VAE	-	Variational Autoencoder

1. CHAPTER - INTRODUCTION

1.1 Problem Background

In the modern world, the Internet has become an integral part of human life. Nowadays, people use computer devices connected to cloud data storage through networking solutions more often, making the Internet even more critical to their regular activities. This is particularly evident when we examine the growth of the Internet over the last two decades. As Figure 1.1 illustrates, the Internet has seen exponential growth in terms of users over the last two decades, as reported by Statista (Statista, 2023).

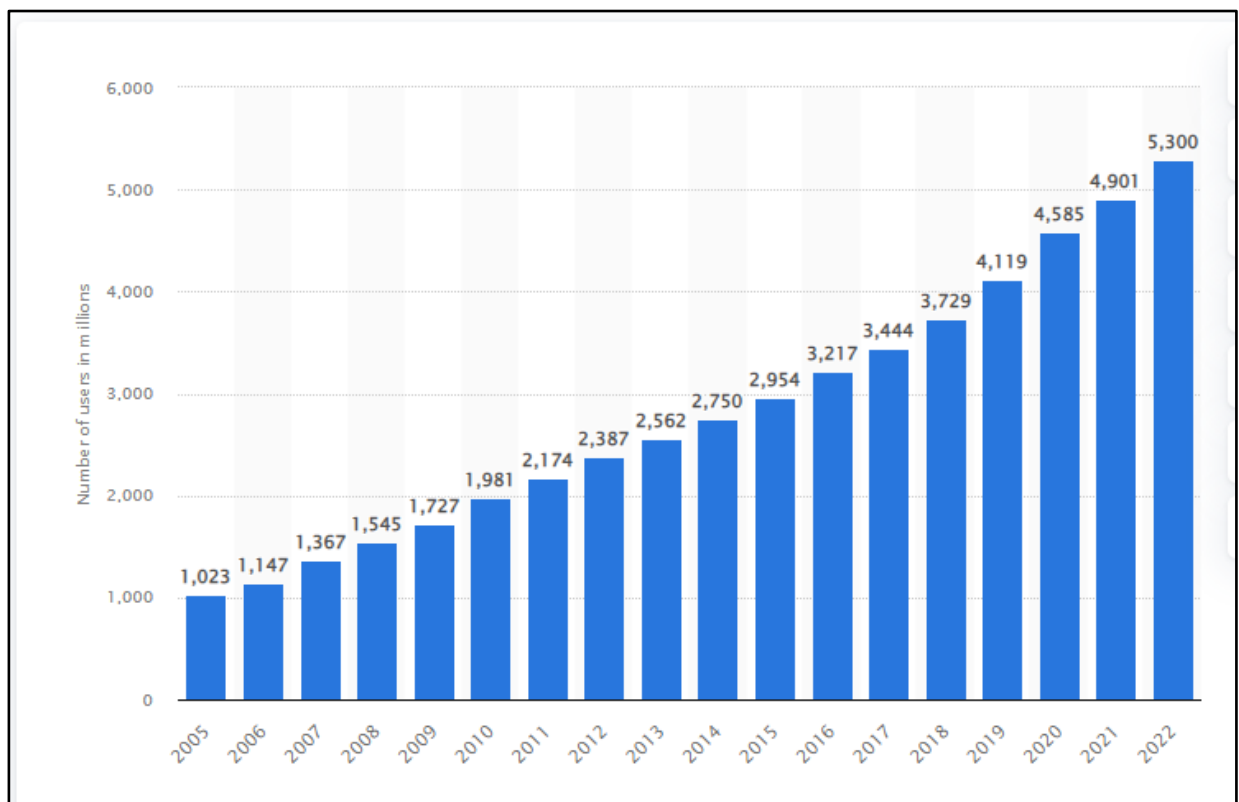


Figure 1.1: Exponential growth of Internet users (statista, 2023)

However, the exponential growth of the Internet has also led to an increase in the number of cyber security attacks that are transmitted through computer networks. Recently, with the emergence of digital currencies, attackers have become more interested in data breaches and the collection of sensitive user data, exploiting network vulnerabilities over computer networks. With the massive increase of cyber security threats, it has become an issue for dealing with those fast growing network threats over the computer world (Savelyeva & Timkina, 2021).

From a network security perspective, signature-based detection is a commonly used method for detecting network attacks. This method relies on a signature database that must be updated frequently to reflect the history of detections. While this method has proven to be quite effective for detecting known attacks, it can be problematic for detecting first-time attacks, as no signature may be found even if the database is up-to-date. Consequently, this approach becomes entirely vulnerable to new attacks, as it takes time for the necessary attack signatures to be populated through the databases. In such cases, damage to the data may already have been done (Li et al., 2017).

Furthermore, most Internet use cases today have been encrypted using Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. Since the usage of SSL/TLS has increased highly on most networking platforms, the ability to observe the encrypted Internet data stream contents has made the network attack detection functions more difficult because SSL/TLS protocols encrypt the traffic between client and server (Radivilova et al., 2018). Hence, signature-based attack detection methods would not work effectively. Since SSL/TLS encryption can be considered an essential security feature, it would be crucial to look for different approaches to network security attack detection which work within the encrypted data streams, as maintaining the balance between data encryption and threat detection is a problem (Radivilova et al., 2018).

1.2 Motivation

The motivation behind this research is rooted in the critical need for robust and adaptive network security systems in our increasingly digital and interconnected world. With the exponential growth of Internet users and the emergence of new cyber threats, ensuring the security of computer networks has become a paramount concern. Traditional network security measures, primarily relying on signature-based detection methods, face significant limitations when dealing with novel and evolving threats. This research is driven by the motivation to overcome these limitations and enhance network security by harnessing the potential of machine learning.

The motivation for this study can be summarized as follows:

Evolving Cyber Threats: The Internet has become an integral part of daily life for people in worldwide. However, this increased connectivity has led to a corresponding increase in cyber threats. The motivation stems from the urgency of addressing these evolving threats effectively.

Limitations of Signature-Based Detection: Traditional security mechanisms predominantly rely on known attack signatures to identify threats. While effective against known attacks, they struggle to detect previously unseen threats and zero-day vulnerabilities. This research is motivated to develop more adaptive and proactive security measures.

Impact of Encryption: With the pervasive use of encryption protocols like SSL/TLS, the ability to inspect network traffic for anomalies has been hampered. The motivation behind this study is to find alternative methods, such as machine learning, to effectively detect anomalies within encrypted traffic.

Machine Learning Advancements: Recent advancements in machine learning, particularly deep learning and ensemble methods, have demonstrated their effectiveness in various applications. The motivation arises from the potential of these techniques to revolutionize network security by improving detection accuracy, scalability, and efficiency.

Real-World Implementation: This research is motivated by the desire to make tangible contributions to real-world network security. By exploring the practical implications of machine learning-based anomaly detection, the study aims to bridge the gap between theoretical research and its actual implementation in network environments.

Overall, the motivation for this research project is to enhance network security in an era of growing cyber threats and network vulnerabilities. It seeks to leverage the power of machine learning to address the shortcomings of existing security measures and contribute to the development of more effective, adaptive, and efficient network anomaly detection systems.

1.3 Statement of the problem

In today's digital world, the Internet is vital. With a growing number of users and devices online, cyber-attacks are rising. Existing security measures work well against known attacks but struggle with new and unfamiliar threats.

The main problem is that these measures rely on updated databases of known attack patterns. These databases contain templates for recognizing attacks but cannot keep up with new threats. By the time they catch up, the damage might have been done.

Additionally, encryptions like SSL and TLS make things more challenging. These security protocols protect data during transmission but make it harder for standard security systems to spot malicious activity.

This research project explores how machine learning can improve anomaly detection in computer networks to address these challenges. Anomalies are deviations from normal network behavior and can indicate cyber-attacks or intrusions. This study explores the possibility of using machine learning to build a more effective anomaly detection system, especially for new and encrypted threats, and provides insights for better network security.

1.4 Research Aims and Objectives

The main aim and the objectives of this research project have been listed in this problem context.

1.4.1 Aim

The proposed research project aims to investigate the effective use of network traffic pattern analysis by utilizing machine learning techniques for detecting and analyzing anomalies in computer networks. Anomalies can be deviations from standard behavior patterns involving malicious activity, such as network intrusions or cyber-attacks.

1.4.2 Objectives

The main objectives of this particular research are as follows:

1. To develop a machine learning-based system for detecting and analyzing anomalies in computer networks.
2. To evaluate the effectiveness of different machine learning algorithms for detecting anomalies in network traffic.
3. To investigate the practical implications of machine learning for anomaly detection in real-world networks.
4. To provide recommendations for improving machine learning-based anomaly detection systems' accuracy, efficiency, and scalability.

1.5 Scope

The project will focus on developing a system that uses machine learning algorithms to detect and analyze anomalies in network traffic automatically. The proposed system will be trained using a network traffic dataset that includes normal and anomalous behavior. The machine learning algorithms will be selected based on their ability to handle large datasets, adapt to changing network conditions, and provide accurate and timely results.

Major Network attack types have targeted the breach of network Accessibility, Confidentiality and integrity. In this project, it has focused on most of the known attacks

types, such as Denial of Service (DoS), Probe (Information Gathering), User to Root (U2R), and Remote to User (R2U) / Remote to Local (R2L) (Roy & Shin, 2019).

The research would be mainly focused on the effectiveness of various machine learning algorithms such as Support Vector Machines (SVM), Random Forests, Naive Bayes, K Nearest Neighbors, Decision Trees, AdaBoost Classifications and Artificial neural networks (ANN). Those algorithms will be evaluated based on their detection accuracy and overall performance.

Furthermore, this research project will investigate the practical implications of utilizing machine learning in real-world networks. It will explore the proposed system in existing network architectures to evaluate the network's scalability, efficiency, and performance under different scenarios.

In this project, we would not try out other anomaly detection methods for the network anomalies, such as human-centric analysis, collaborative detection, heuristic-based detection, and rule-based systems. It would focus more on ML-based approaches that utilize diverse methods such as statistical analysis, flow analysis, and deep packet inspection (DPI) (Smith, J & Johnson, A (2023).

1.6 Significance of the Research

The significance of this research project lies in its potential contribution to the field of cyber security. With the continuous growth of digital networks and the increasing sophistication of cyber threats, there is a pressing need for innovative approaches to network security. The outcomes of this research could solve these challenges by providing a more advanced and adaptive means of detecting network anomalies.

Moreover, the findings and recommendations of this study may benefit various stakeholders, including cybersecurity professionals, organizations, and institutions. Improved network anomaly detection systems could aid in mitigating the impact of cyber-attacks, safeguarding sensitive data, and ensuring the uninterrupted operation of critical infrastructure. Ultimately, the research aims to strengthen the security posture of digital networks, offering a proactive defense against emerging threats.

1.7 Structure of the Thesis

The main chapters and content of those chapters would be as follows.

- **CHAPTER 1: INTRODUCTION**

Chapter 1 introduces the research's motivation, problem statement, objectives, and scope and outlines the thesis structure.

- **CHAPTER 2: LITERATURE REVIEW**

In Chapter 2, we delve into existing research and theories relevant to our study, setting the foundation for our research.

- **CHAPTER 3: METHODOLOGY**

Chapter 3 outlines our research methods, including the design, data collection, and analysis techniques employed in our study.

- **CHAPTER 4: EVALUATION AND RESULTS**

Chapter 4 presents the findings and results of our research, supported by data analysis.

- **CHAPTER 5: CONCLUSION AND FUTURE WORK**

The final chapter summarizes our findings, discusses their implications, and suggests for future research directions.

2. CHAPTER - LITERATURE REVIEW

2.1 Introduction

This literature review study focuses on the existing research to explore anomaly detection using traffic pattern analysis based on machine learning. We start by emphasizing the significance of this literature review in anomaly detection and machine learning. Here, we lay the preliminary platform for understanding how the existing body of knowledge has shaped the field, making it essential for our research to build upon these foundations.

Next, the theories and concepts form the theoretical backdrop of our research, showing how these frameworks have influenced the development of anomaly detection methods in machine learning.

Nevertheless, before we leap forward, we rewind to explore the historical evolution of anomaly detection based on traffic patterns and machine learning within the research landscape.

Empirical studies would be invaluable to us prior to our research, focusing on their methodologies, successes, and shortcomings in the context of anomaly detection.

Research methods matter, particularly in a dynamic field like anomaly detection. We will assess the tools employed by past researchers, extracting insights on their effectiveness and potential areas for refinement.

Lastly, it would focus on the gaps, those intriguing spaces where questions remain unanswered and anomalies unresolved. These gaps would motivate future researchers, guiding them toward the unknown territory that particular anomaly detection based on traffic pattern analysis study will contribute.

2.2 Theoretical Framework

In the forthcoming section, it has ventured on a journey through the theoretical framework that serves as the backbone of this particular research. Goal is to take a deep dive into the core principles that steer the approach to anomaly detection, especially within the ever-evolving domain of network security. The objective here is to offer a thorough awareness of the fundamental theories and ideas that have carved the anomaly detection, while emphasizing their direct applicability to identifying network threats.

2.2.2 Common Anomalies in Network Security

To ensure minimal overlap in standardization related to Information/Network Security, it's essential to maintain a comprehensive catalog of standardization activities and establish a method for referencing standards. This approach allows for efficient tracking of the impacts of changes in dependencies.

Cyber security encompasses the defense of computers, servers, mobile devices, electronic systems, networks, and data against malicious attacks. This broader field, also known as IT security or electronic data security, includes aspects of computer security, disaster recovery, and user awareness. Cyber security safeguards against three primary threats: cybercrime, involving financial gain or gang-based attacks; cyber war, often linked to information gathering and politically motivated activities; and cyber terrorism, aimed at undermining electronic systems to create panic and fear (Cisco, 2017).

Cyber security pertains to the security of cyberspace, referring to the relationships among the interconnected links and objects accessible through telecommunications networks. These objects serve as interfaces allowing system control, remote data access, and integrated participation within the Cyberspace. In figure 2.1, it shows various areas in Cyber Security (ECS, 2020).

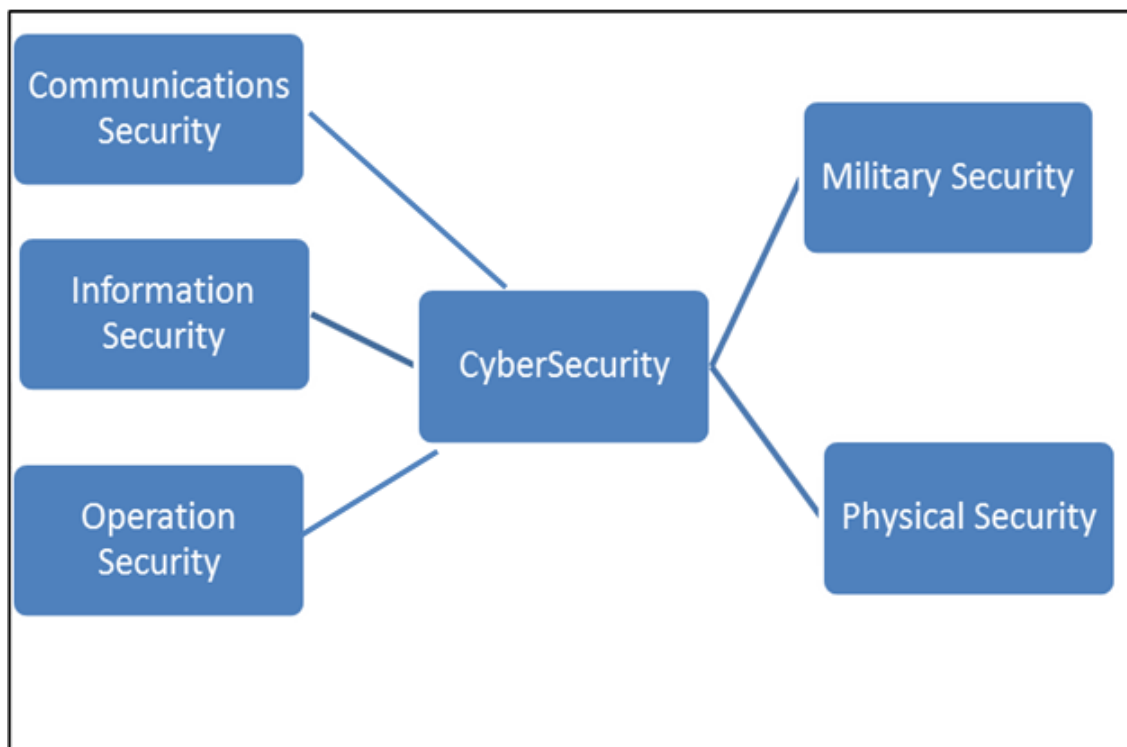


Figure 2.1: Areas in cyber security (ECS, 2020)

Multiple domains within cyber security exist, as shown in Figure 2.1. Human and technical factors significantly impact the security of sensitive information in cyberspace. In the following sections, we'll explore these factors in detail.

The rapid evolution of technology has often outpaced human adaptability, leading to errors in organizations and technology. In organizational settings, the safety and performance of employees are closely intertwined. Companies dealing with human factors must ensure the well-being and health of their workforce (Cisco, 2017).

Most Common Types of Network Attacks

A network attack is any offensive action aimed at altering, destroying, or stealing sensitive information from computer-based systems and devices. To safeguard against these attacks, individuals and organizations should be aware of the most common types that can occur in cyberspace. In this section, we will describe these prevalent cyber-attacks.

Denial-of-Service (DoS) Attacks

DoS attacks overwhelm system resources to the point where the system can't respond to service requests. These attacks are executed through a vast number of machines controlled by malicious software under the attacker's command (Alpine, 2020). The primary goal is to

disrupt services, gain a competitive edge, or launch additional attacks while the system is offline. Notable DoS attack types include teardrop attacks, TCP SYN flood attacks, ping-of-death attacks, smurf attacks, and botnets.

Man-in-the-Middle (MitM) Attacks

In MitM attacks, an attacker intercepts communication between a client and a server. A common form is session hijacking, in which the attacker establishes a session between the client and the server. This manipulates the server into believing it's still communicating with the trusted client. IP spoofing is another MitM method, where an attacker appears as a trusted partner (Alpine, 2020). These attacks aim to exploit vulnerabilities in the communication process. To defend against MitM attacks, digital certificates, and encryption methods offer effective protection. Figure 2.2 shows the MitM attack process (Ghosh, A., & Senthilrajan, A., 2020).

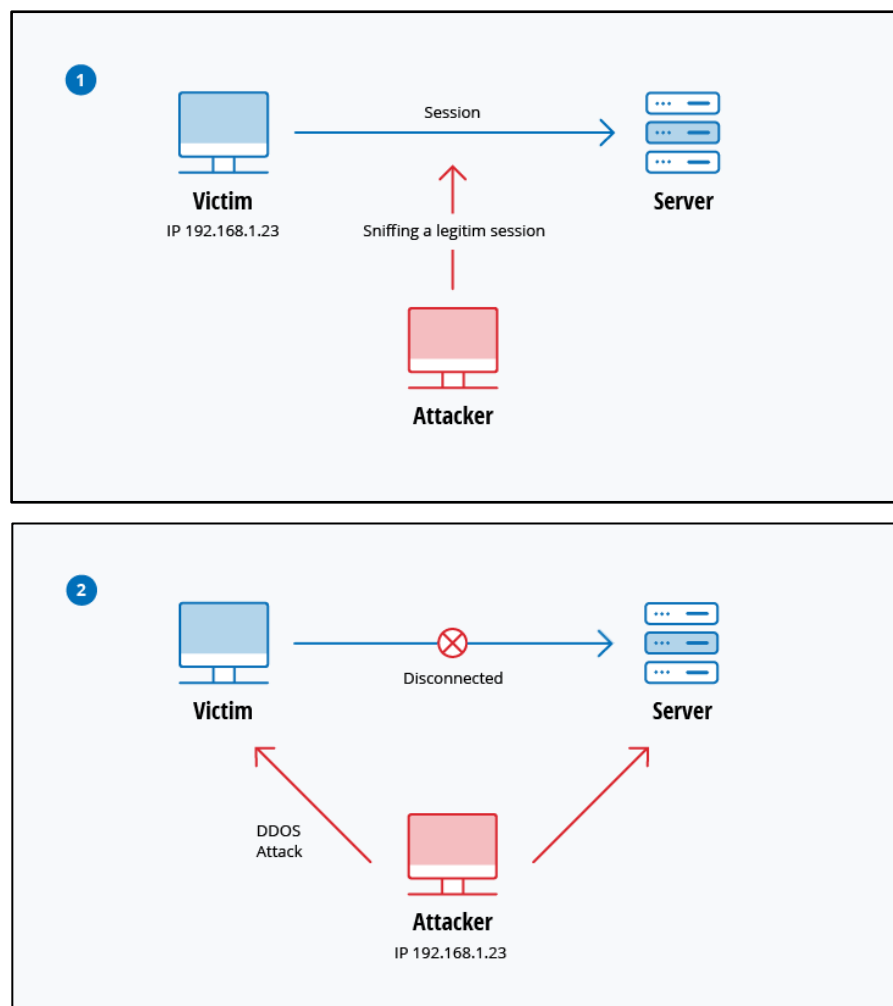


Figure 2.2: Procedure of session hijacking (Ghosh, A., & Senthilrajan, A., 2020)

Phishing Attacks

Phishing attacks involve sending emails that appear to come from trusted sources. The goal is to gain sensitive personal information by manipulating individuals into taking specific actions. Attackers often use social engineering and technical tricks to deceive recipients. The most sophisticated form is spear-phishing, in which attacker's research victims to create highly relevant and personal messages. Combating phishing attacks requires critical thinking when handling emails, validating email headers, and hovering over links before clicking.

Drive-by Attacks

Drive-by download attacks commonly spread malware. Attackers search for insecure web applications, embedding malicious scripts into web pages. When users visit these sites, the malicious script installs malware directly on their computers. Unlike many other attacks, drive-by attacks don't require user interaction and can exploit vulnerabilities in trusted applications or browsers (Rapid7, 2017). Protecting against these attacks involves keeping operating systems and web browsers up to date, enabling security policies, and sticking to trusted websites.

SQL Injection Attacks

SQL injection attacks target database-driven applications. Attackers execute SQL queries within the database using client-to-server data. This allows them to manipulate data, inject commands, or perform actions, including dropping databases or recovering data files (Rapid7, 2017). To prevent SQL injection attacks, it's crucial to understand potential vulnerabilities, apply advanced user privilege models, use stored procedures, and validate data input.

Password Attacks

Password attacks are the easiest way to gain unauthorized access. Attackers steal passwords, often using methods like observing users' actions, network sniffing, or social engineering (Rao et al., 2015). Techniques like brute-force attacks, where passwords are randomly guessed, and dictionary attacks, where common passwords are tried, can increase the probability of success. To counter these attacks, account locking after a few incorrect attempts

Cross-site scripting attack (XSS)

Cross-site scripting (XSS) attacks involve exploiting vulnerabilities in third-party web applications to inject malicious scripts into a user's browser. Attackers insert a payload containing a harmful script into the web application's database. When a victim requests a page from the website, the attacker's payload is sent to the user's browser within the page's HTML content. Once executed, the script can lead to various detrimental consequences, including stealing the victim's cookies for session hijacking. XSS attacks can also be used to exploit further vulnerabilities, such as logging keystrokes, capturing screenshots, discovering network information, and gaining control of the victim's device (Mahrouqi, Tobin, Abdalla, & Kechadi, 2016). Figure 2.3 has depicted the cross-site scripting attack process (Singh and Jain, 2018).

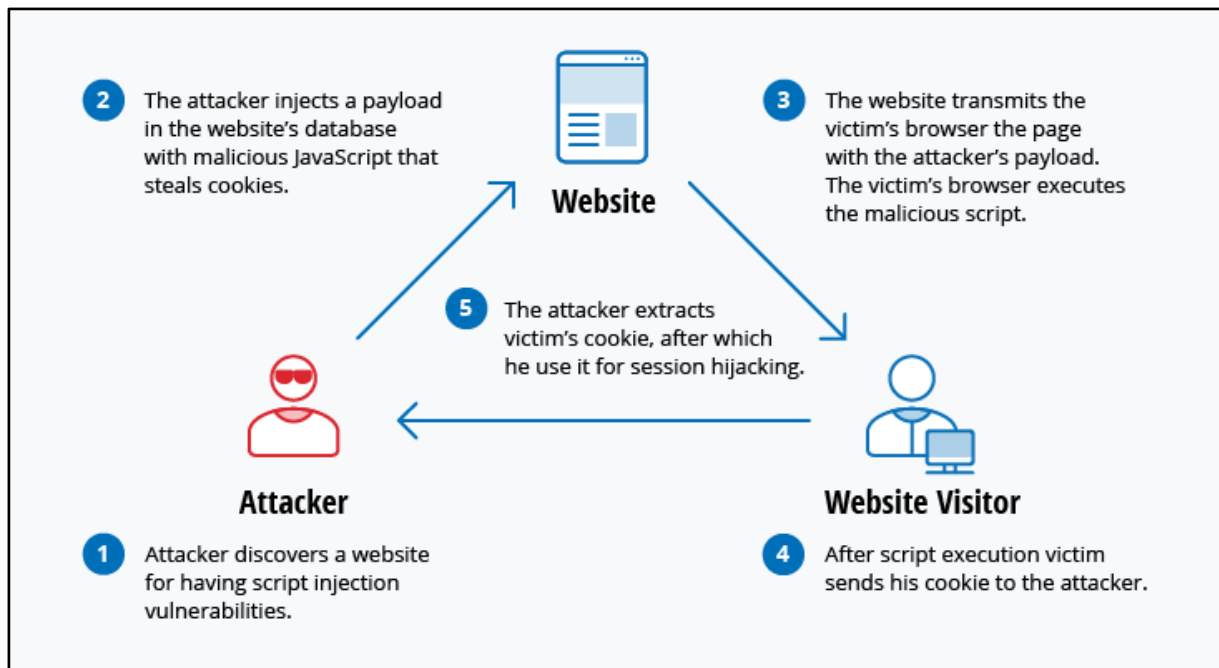


Figure 2.3: Cross-site scripting attack (Singh and Jain, 2018)

XSS attacks can leverage various scripting languages, including VBScript, Flash, and ActiveX. JavaScript is commonly abused due to its wide support in web environments. To defend against XSS attacks, developers should thoroughly sanitize user input in HTTP requests before rendering it. All data must be filtered, escaped, or validated before being displayed to users, particularly query parameters during searches. Special characters such as `'`, `<`, `>`, `?`, and `&` should be converted into their respective HTML-encoded equivalents. It's advisable to allow end-users to disable client-side scripts if possible (Singh & Jain, 2018).

Eavesdropping Attacks

Eavesdropping attacks involve intercepting network traffic to steal sensitive user data, including passwords and credit card details. These attacks can be either passive or active in nature (Alpine, 2020).

Passive eavesdropping involves attackers listening to data transmissions across the network to detect and capture data. In contrast, active eavesdropping sees attackers posing as friendly entities, sending queries to transmitters to obtain information. These actions are often referred to as scanning, probing, or tampering (Alpine, 2020).

Detecting and preventing passive eavesdropping is more critical than countering active attacks. Data encryption provides an effective defense against eavesdropping.

Malware Attacks

Malware attacks involve the installation of unwanted software on a system without the user's consent. Malware can embed itself in legitimate software or propagate through communication networks. It may hide within useful applications or self-replicate via the Internet (Rao et al., 2015).

Various types of malicious tools are used in malware attacks, including macro viruses, file infectors, system or boot record infectors, polymorphic viruses, stealth viruses, trojans, logic bombs, worms, droppers, ransomware, adware, and spyware (Rao et al., 2015).

Birthday Attacks

Birthday attacks target hash algorithms used to verify message integrity, digital signatures, or software. These attacks exploit the possibility of finding two random inputs that generate identical message digests (MDs) when passed through the hash function. Once an attacker identifies the same MD for the message in use, they can replace the user's message without detection (Singh & Jain, 2018). Data encryption is the most effective countermeasure against birthday attacks.

2.2.4 Machine Learning and Anomaly Detection

In this section, it would be focused on the machine learning concepts in the context of anomalies detection.

Statistical Models in Anomaly Detection

The history of statistical models in anomaly detection is strong and influential, providing structured methodologies for understanding deviations from established benchmarks (Smith, 2020). In the realm of network security, these models have played a key role in identifying irregular patterns and behaviors that may signal potential attacks. By establishing a baseline for what constitutes 'normal' behavior within a network, statistical models can trigger alerts when deviations from this baseline occur, enabling security professionals to investigate potential threats (Johnson, 2019). The historical significance of statistical models in network security is further underscored by their capacity to capture a wide range of network threats, including intrusion attempts and distributed denial-of-service (DDoS) attacks.

Machine Learning Algorithms for Anomaly Detection

The realm of machine learning offers a diverse array of algorithms and techniques that hold great promise in the detection of network threats (Mobilio et al., 2019). From decision trees to support vector machines, machine learning methods can adapt to the ever-evolving threat landscape of the digital realm (Moustafa et al., 2017). Decision trees provide a structured approach to decision-making, making them suitable for classifying network behaviors as normal or anomalous. Support vector machines excel in identifying complex patterns within network data. These algorithms can analyze network data patterns, identify deviations, and provide early warnings in response to potential threats.

Data-Driven Approaches

Data-driven approaches, such as clustering and dimensionality reduction, offer a unique perspective on understanding network threats (Osanaiye et al., 2016). These techniques, often employed with large volumes of network data, excel at uncovering modest patterns that may not be apparent through traditional analysis methods. Clustering techniques, like K-Means, group network data into clusters based on similarities, revealing patterns that may indicate

threats. Dimensionality reduction methods, such as Principal Component Analysis (PCA), simplify complex data for easier anomaly identification. These techniques can efficiently detect complex threats within network traffic, facilitating the discovery of patterns indicative of emerging attacks.

Network Threats and Anomaly Detection

Learning the complexities of network threats is fundamental in the context of anomaly detection (Umer et al., 2017). Network security professionals must be well-versed in the landscape of threats, including intrusion attempts, malware infections, denial-of-service attacks, and other cyber threats. These threats are dynamic and ever-evolving, presenting unique challenges to network security. Intrusion attempts may involve attackers trying to gain unauthorized access to a network, while malware infections can lead to the compromise of network resources. Denial-of-service attacks disrupt network services, causing downtime and financial losses. Anomaly detection methods are equipped to address these threats effectively by identifying patterns and behaviors that deviate from the norm.

Challenges and Limitations

While theoretical frameworks provide a solid foundation for anomaly detection, they are not without challenges and limitations (Nisioti et al., 2018). It is crucial to acknowledge these issues to enhance the effectiveness of anomaly detection methods. The persistence of false positives remains a challenge, as anomaly detection systems may sometimes generate alerts for behaviors that are not actual threats. Additionally, the ever-evolving nature of threats necessitates regular updates to detection methods and algorithms to remain effective. Real-time detection is essential to respond promptly to threats.

Bridging Theories and Practical Application

Bridging the gap between theory and practice is pivotal in the context of network threat detection (Moustafa et al., 2017). Theoretical insights must be translated into practical applications to build robust anomaly detection systems. Implementing theoretical knowledge is crucial to address real-world security challenges effectively. The transition from theory to practice involves the design and deployment of anomaly detection systems, where the insights gained from the theoretical framework find their real-world utility.

This comprehensive exploration of the theoretical framework emphasizes the relevance of various theories and concepts in the context of network threat detection. It establishes the stage for our research, which aims to innovate and advance anomaly detection within the dynamic domain of network security, addressing the unique challenges posed by diverse and evolving network threats.

2.3 Related Work

This related work section of the literature review study aims to summarize some of the latest research related to the particular research context. Network anomaly-based threat detection mechanisms play a crucial role in ensuring the reliability and security of computer networks. Recently, there has been a growing interest in using machine learning and other network behavior analysis techniques to detect network anomalies, as opposed to conventional methods focusing on directly identifying security threats. It has identified supervised or semi-supervised machine learning models as the most widely used techniques for behavior-based anomaly detection in computer network attack detection. Support vector machines, neural networks, and decision tree models were widely used here. In those models, labeled instances were required to train on a dataset on anomalous or normal network traffic.

A study conducted by Vivekanandan and Praveena (Vivekanandan, K., & Praveena, N., 2020) suggested a hybrid deep learning model that combines long short-term memory (LSTM) and convolutional neural networks (CNNs) networks in order to identify anomalies behaviors in a network according to the semi-supervised learning technique. In that study, high accuracy was based on various benchmark datasets, including NSL-KDD and Cup 99.

Another significant learning approach would be based on unsupervised learning. In this approach, the learning model is trained to detect anomalies without having explicit labeling prior to the experiment. This approach could be utilized in situations in which labeled data would be infrequent or hard to obtain. An example study made by Yao (Yao, R., et al., 2019) suggested that an unsupervised algorithm in anomaly detection is based on a deep generative model called a Variational Autoencoder(VAE) model. In that VAE, it was trained based on a massive dataset of normal traffic, and it was able to identify network anomalies by figuring out the reconstruction error of network packets in incoming mode.

The ensemble learning model has been identified as another approach used with a combination of multiple base models to improve the accuracy of the learning system overall. An example study conducted by Yang et al. (Yang et al., 2020) proposed that an ensemble approach has combined the classifiers in a multiple manner, including decision trees, gradient boosting machines, and random forests, to detect anomalies in computer networks. They were able to achieve great accuracy on several known datasets, including the CICIDS 2017 and NSL-KDD datasets.

Fernandez and Xu (Fernandez, A., & Xu, L , 2018) conducted a case study where they harnessed the capabilities of a deep-learning network to detect anomalies in network traffic. Their study primarily focused on supervised network intrusion detection, a domain where their approach exhibited remarkable success. Notably, their research shed light on the remarkable efficiency achieved by considering only the initial three octets of IP addresses, a strategy to address the dynamic nature of IP allocation, particularly in the context of Dynamic Host Configuration Protocol (DHCP). This approach underscored the unique effectiveness of Deep Neural Networks (DNN), especially in handling anomalies related to DHCP. Furthermore, Fernandez and Xu's work illustrated the potential of autoencoders in anomaly detection, particularly when trained on expected network flow patterns.

In a similar vein, Kwon (2019) proposed the utilization of Recurrent Neural Networks (RNN) and Deep Neural Networks (DNN) in tandem with machine learning techniques for network anomaly detection. Kwon's study was complemented by local experiments that affirmed the viability of the DNN approach in the analysis of network flow traffic. Within this research, there was a comprehensive exploration of the effectiveness of DNN models in the analysis of network flow traffic, further augmented by experiments involving their Fully Convolutional Network (FCN) model. The findings of these experiments were promising, revealing that the DNN approach surpassed traditional machine learning methods such as Support Vector Machines (SVM), Random Forest, and Adaboost in terms of accuracy when it came to detecting anomalies.

Moving on, Garg and colleagues (2020) introduced a novel hybrid data processing model intended for the detection of anomalies in network traffic. This innovative model integrated Grey Wolf Optimization (GWO) with Convolutional Neural Networks (CNN). The researchers significantly improved the training techniques for both GWO and CNN, enhancing their capabilities in exploration and the capture of initial population states, leading to the development of what they termed "Improved-GWO" and "Improved CNN." The proposed model operated in two distinct stages. In the initial stage, Improved-GWO was employed for the purpose of feature selection, aiming to strike a balance that would reduce false positive rates while minimizing the feature set. Subsequently, in the second stage, Improved CNN came into play, facilitating the classification of network anomalies. The authors evaluated the model's efficiency using benchmark datasets (DARPA'98 and KDD'99) alongside artificial datasets. The results were compelling, clearly demonstrating the

superiority of the proposed cloud-based anomaly detection model in comparison to other existing approaches. The model exhibited significant improvements, including an 8.25% increase in detection rate, a 4.08% decrease in false positives, and a 3.62% boost in accuracy when measured against the standard GWO combined with CNN.

In most recent studies, there was growing interest in using deep learning models more often for network anomaly detection. Trained models such as convolutional neural networks (CNNs) and deep neural networks (DNNs) were more common between them. A study conducted by Liu et al. (2021) has proposed a model called DeepLog, which is a deep learning model that uses deep neural networks (DNNs) to recognize the network log patterns in normal traffic in order to identify network anomalies by observing the deviations out of these network traffic patterns. They were able to achieve high accuracy on various benchmark datasets, such as CICIDS 2017 and UNSW-NB15 (Verma & Ranga, 2019).

A newer study explored by Kopčan et al. (Kopčan et al. 2021) Uses generative models, such as generative adversarial networks and variational autoencoders, to detect anomalies in network traffic. This study achieved excellent results, with reported detection errors of 0.08% on an AAE model and 1.89% on a GAN model trained on three datasets: MNIST, CIFAR-10, and Fashion-MNIST.

Another approach to network anomaly detection using machine learning was the development of a novel deep learning algorithm that could identify anomalies with high accuracy in a manner that also supports explaining its decision-making process in a more transparent manner rather than just not only using a hidden automatic detection manner (Balasubramaniam & Arnon, 2021). In such a case, the algorithm could be a combination of recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention mechanisms that would help to detect anomalies in network traffic patterns (Alsulaiman & Al-Ahmadi, 2021).

Moreover, such an algorithm could be incorporated with explainable AI (XAI) techniques. In this case, it would generate saliency maps and heat maps like visualization instances to provide informative insights into the patterns and features that could be important in detecting network anomalies.

Such an approach could address the challenge of explaining ability in machine learning-based anomaly detection systems, which is often a concern in critical applications such as cyber security. By providing clear and interpretable explanations for its decisions, the algorithm could increase trust in the system and facilitate more effective collaboration between human operators and automated detection tools.

Mobilio et al. (2017) introduced a fascinating concept in anomaly detection by presenting a Cloud-based service. They took it a step further by demonstrating early results with lightweight detectors, which promised better control over anomaly detection logic. They talked about how this paradigm could be applied to anomaly detection and offered insights into achieving "anomaly detection as a service." They even proposed an architecture that could work alongside any observation system storing data in time-series databases to support this process. Their experiments, particularly those carried out with the Clearwater cloud system, shed light on how the "as-a-service" paradigm can effectively manage anomaly detection logic into the integration of this new technology into real-time anomaly detection.

Moustafa et al. (2018) took an exciting journey by introducing a Collaborative Anomaly Detection Framework (CADF) designed to tackle the challenges posed by large-scale data in cloud computing systems. Their framework is composed of three key modules. First, they looked at capturing and logging network data and then preprocessing it to make it more usable for analysis. However, their innovative decision engine, which utilizes a Gaussian Mixture Model and a lower-upper Interquartile Range threshold for identifying attacks, was more critical. What is impressive is their choice of dataset - the UNSW-NB15 - which they used to evaluate the Decision Engine's reliability, especially in a real cloud computing environment. They also compared their framework with three Anomaly Detection Systems (ADS) techniques. To make it even more accessible, they developed Software as Service (SaaS) architecture, making it easier to deploy in the cloud.

Osanaiye et al. (2019) presented an approach that turned out to be a game-changer in cloud computing. They introduced an ensemble-based multi-filter feature selection method that's efficient and immensely practical. This method stands out because it combines the output of four filter methods to achieve the best possible feature selection. Its application in cloud computing in detecting Distributed Denial of Service (DDOS) attacks was crucial. Their extensive experimentation, utilizing the well-known NSL-KDD intrusion detection dataset, revealed that their method significantly reduced the number of features needed for effective

detection. The impressive results show a high detection rate and accuracy compared to other classification techniques.

Barbhuiya et al. (2016) proposed a Real-time Anomaly Detection System (RADS), a promising solution for dealing with anomalies. What is unique about RADS is its approach to detecting anomalies using a single-class classification model and a window-based time series analysis. In a lab-based environment within an OpenStack-based Cloud data center, they evaluated its performance and, interestingly, in real-world scenarios within a Cloud data center named Bitbrains. The results are significant because RADS achieved an impressive accuracy rate of 90-95% with a meager false-positive rate (0-3%) when detecting DDoS and crypto-mining attacks in real-time. Also, RADS proved to be a lightweight tool that did not hog the system resources, making it practical for use in a Cloud data center.

Zhang (2019) delved into Multi-view learning techniques for cloud anomaly detection, which is quite an advanced topic. What is particularly interesting is how this technique handles the complexities of anomaly detection in cloud computing. It automatically integrates features from different subsystems, improving classification solutions by minimizing training errors. Additionally, it uses weighted samples to retrain the classification model, which is an intelligent approach. This model tackles several challenges, such as imbalanced data and high-dimensional features, effectively using Multi-view learning and feed regulating. It is a testament to the ever-evolving field of anomaly detection in cloud computing.

Umer et al. (2018) introduced a fascinating approach to intrusion detection systems (IDS). Their focus was on flow-based IDS that operate on IPFIX/NetFlow records, which are treated as input. These records come with many attributes, and the team made smart choices about which ones to employ. Attributes like the originating IP address and destination port took center stage in their detection approach. They conducted feature selection for the relevant attributes to enhance the system's performance. Furthermore, they encountered a preprocessing phase for flow records to ensure they aligned with the requirements of anomaly detection algorithms.

Nisioti et al. (2017) embarked on a survey of unsupervised models for IDS systems. The uniqueness of these models lies in their ability to extract features from diverse data sources, including network traffic and logs from various devices and host machines. Their flexibility in handling additional features from diverse sources without requiring recurrent training was

noticeable. The team also delved into feature selection methods for IDS, aiming to find the optimal feature subsets for each class. This approach reduces computational complexity.

Münz et al. (2016) introduced a novel detection model focused on identifying anomalies in network traffic. What is intriguing is their utilization of a clustering algorithm, K-Means, for the input data. Their approach involves capturing hypervisor packets and arranging them into a sequence of packet flows aligned with the operating system's timeline. The model operates in two phases, with the first phase focusing on feature extraction from the packet headers, creating a primary feature vector for each unique packet. The second phase involves extracting a separate feature vector for each packet flow associated with the primary feature vectors of the whole packet.

Aldribi et al. (2019) ventured into the realm of hypervisor-based cloud IDS, offering a novel feature extraction technique grounded in user instance activities and their behaviors within the hypervisor. Their primary goal was to identify anomalous behaviors in the cloud environment by tracking statistical variations. They employed a combination of gradient descent algorithms and E-Div to achieve this. They introduce a new intrusion detection dataset collected in a cloud environment that is publicly available for researchers. This dataset includes multistage attack scenarios, enabling the development and evaluation of threat environments in cloud computing. They conducted experimental evaluations, employing the Riemann rolling feature extraction scheme, producing promising results. Of particular interest is the dataset's coverage of communications over encrypted channels, such as those involving protocols like SSH.

Traditional machine learning algorithms have shown promising outcomes in network anomaly detection. However, they might not be able to capture some of the relationships and complex patterns within the data, resulting in the security threat detection in networks not being as efficient as expected. On the other hand, deep learning has shown the potential to discover more complex relationships out of the raw data, resulting in much better performance (Narayan & Shanmugapriya D., 2022). In addition, incorporating practices such as reinforcement learning and unsupervised learning could explore novel approaches for network security threat detection. Hence, these techniques have proven to improve the capability of a system to adapt to dynamic network environments to improve the detection of unseen anomalies.

2.4 Summary

In conclusion, network anomaly detection based on traffic pattern analyzing techniques is still a very active area of research since many approaches and methods are still being explored in a trending manner. Supervised and semi-supervised learning models are popular, but unsupervised, ensemble, and deep learning models are also gaining attraction. Further research is needed to determine the most effective approach for network environments and attack scenarios.

Limited data availability is a significant challenge for anomaly detection using machine learning, as anomalies are rare events. Imbalanced data is another issue, as the small fraction of data representing anomalies can mislead results. Scalability is another challenge, as ML models can be computationally expensive. Lack of explainability and trustworthiness is a significant issue, and the vulnerability of Machine learning models to adversarial attacks is also a significant challenge.

After elaborating on the limitations and gaps of existing research, gaps such as non-up-to-date datasets used heavily due to the experimental issues and cost factors, lack of diversity of attack types used for the testing, and a biased selection of ML models have been identified. This study will address a few key areas that need improvement in previous research studies. A novel approach will be attempted by evaluating more machine learning techniques with up-to-date and highly classified datasets from various attack types to strengthen the proposed system for network anomaly detection. This approach has been selected, assuming it will improve the system's accuracy.

These limitations suggest that more research is needed to develop scalable and robust traffic analysis-based approaches for network anomaly detection.

3. CHAPTER - METHODOLOGY

In this chapter, we aim to provide a comprehensive overview of the proposed methodology for our research. We will commence by explaining the research design approach and offering insights into how the research will be structured and executed. Furthermore, we will delve into the vital sampling techniques and detail the fact-finding methods utilized. We aim to outline the research workflow, emphasizing our commitment to gathering information to support our study's objectives.

3.1 Research Strategy and Ethics

The research strategy was carefully structured to ensure a smooth progression from one phase to the next while minimizing potential delays in achieving project milestones (Saunders et al., 2009). A Work Breakdown Structure (WBS) was implemented to maintain project timelines and prevent deviation from the designated time intervals. Utilizing a Gantt chart was pivotal in this research, visually representing research milestones. Gantt charts allowed researchers to follow a predefined sequence of tasks and ensure they adhered to the project schedule, reducing conflicts among interdependent tasks (Saunders et al., 2009).

Given the sensitive nature of the data, the research design was important, supporting sound decision-making processes. Additionally, the research planned to maintain the utmost professionalism and privacy when collecting data through sources, upholding the integrity of the observed information.

In the realm of ethics and research access (Saunders et al., 2009) strict adherence to appropriate methods for data collection was necessary. Utilizing literature resources was done with the utmost care, ensuring proper acknowledgment and refraining from using any sources not permitted for academic purposes. Moreover, all forms of discrimination were strongly discouraged from all angles, reinforcing the commitment to maintaining ethical research practices throughout the study. This dedication to professional standards has been a fundamental aspect of the research process from its inception.

3.2 Research Design

The research has used the following methodology:

1. Conduct a comprehensive literature review on different machine learning that has been used as algorithms for anomaly detection in network traffic.
2. Use a known large dataset that has covered most known attacks and newer attacks using that simulate the network traffic that includes both anomalous and normal behavior.
3. Data preprocessing, removing noise and irrelevant data would be necessary as the preprocessing steps. Training and evaluating the different learning algorithms on the dataset would be followed to select a suitable algorithm.
4. Implement the selected algorithm in a network environment and evaluate its performance.
5. Finally, it would analyze the results to provide practical recommendations for improving machine learning-based network anomaly detection systems' efficiency, accuracy, and scalability.

In order to have a large and comprehensive dataset, it has analyzed well-known datasets with up-to-date network traffic information. The reason for choosing external datasets is that it can be challenging to obtain a diverse range of network traffic on a public or private network, and it could also raise privacy and security concerns for our network. However, some malicious traffic modeling tools may be used per research requirements.

The following major Research Hypothesis has been identified and will be tested within the research.

Hypothesis 1: A traffic pattern analysis-based network anomaly detection system outperforms rule-based methods in accuracy and effectiveness.

Hypothesis 2: Supervised and unsupervised algorithms vary in detecting network anomalies within a supervised dataset, with potential differences in accuracy and efficiency.

Hypothesis 3: Machine learning applied to network anomaly detection provides insights into intrusions, enhancing proactive cyber security.

Hypothesis 4: Advanced machine learning techniques optimize scalability, accuracy, and efficiency of anomaly detection systems.

Hypothesis 5: Blending historical data and real-time metrics in a machine learning framework enhances network adaptability and robustness against emerging threats.

In Figure 3.1, it shows the research design as a flow chart.

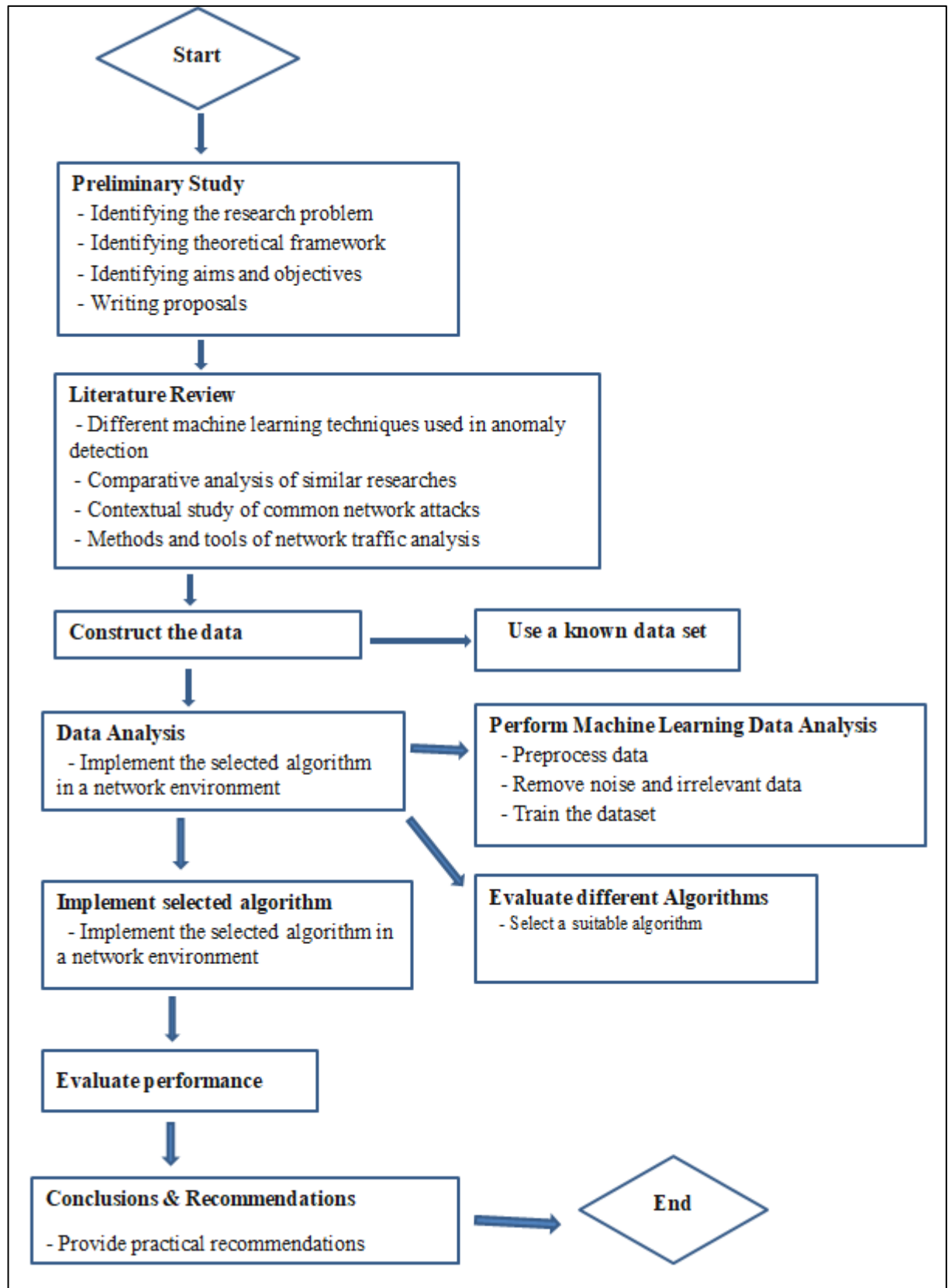


Figure 3.1: Research design flow chart

3.3 Research Evaluation process

In this section, it has provided detailed evolutionary steps that cover all major research activities.

3.3.1 Evaluation Metrics and Measures

Based on the anomaly detection model, we expect the following results: TP-True positive, FP-False positive, FN-False negative, and TN-True negative. Based on these results, the model's outcome will be evaluated using the following metrics (Ghoneim, 2019).

1. **Accuracy** defined as the ratio of the successfully categorized data out of the total data (Ghoneim, 2019). Hence,

$$\text{Accuracy} = (TN+TP) / (FP+TN+TP+FN)$$

2. **Recall** defined as the ratio of data which will be classified as an attack to all attack data (Ghoneim, 2019)

$$\text{Recall} = TP / (TP+FN)$$

3. **Precision** is defined as the ratio of successful classified data as the attack out of all data classified as attacks (Ghoneim, 2019).

$$\text{Precision} = TP / (FP+TP)$$

4. **F-measure** has been defined as the harmonic-mean of the precision and sensitivity (Ghoneim, 2019).

$$\text{F-Measure} = 2 / (1 / \text{Recall} + 1 / \text{Precision})$$

5. **AUC-ROC Score:** AUC ROC, which stands for "Area under the Curve" of the "Receiver Operating Characteristic" curve, serves as a metric to evaluate the performance of a machine learning model. It gauges the binary classifier's capability to differentiate between classes and functions as a concise summary of the ROC curve.

6. **Confusion matrix** is a table used in classification to evaluate the performance of a machine learning model. It provides a summary of the classification results and is particularly useful when dealing with imbalanced datasets. The confusion matrix consists of four metrics:

3.3.2 Comparison with Existing Solutions

By leveraging the evaluation strategy beyond the immediate research scope, it has focused on analytical comparison with the existing research solutions in the context of network anomaly detection. By benchmarking and analyzing proposed models against well-established methods, we seek to identify the unique innovations, potential avenues, and strengths for improving the discovered approach more comprehensively.

Figure 3.2 shows the research evolution process in a flow chart.

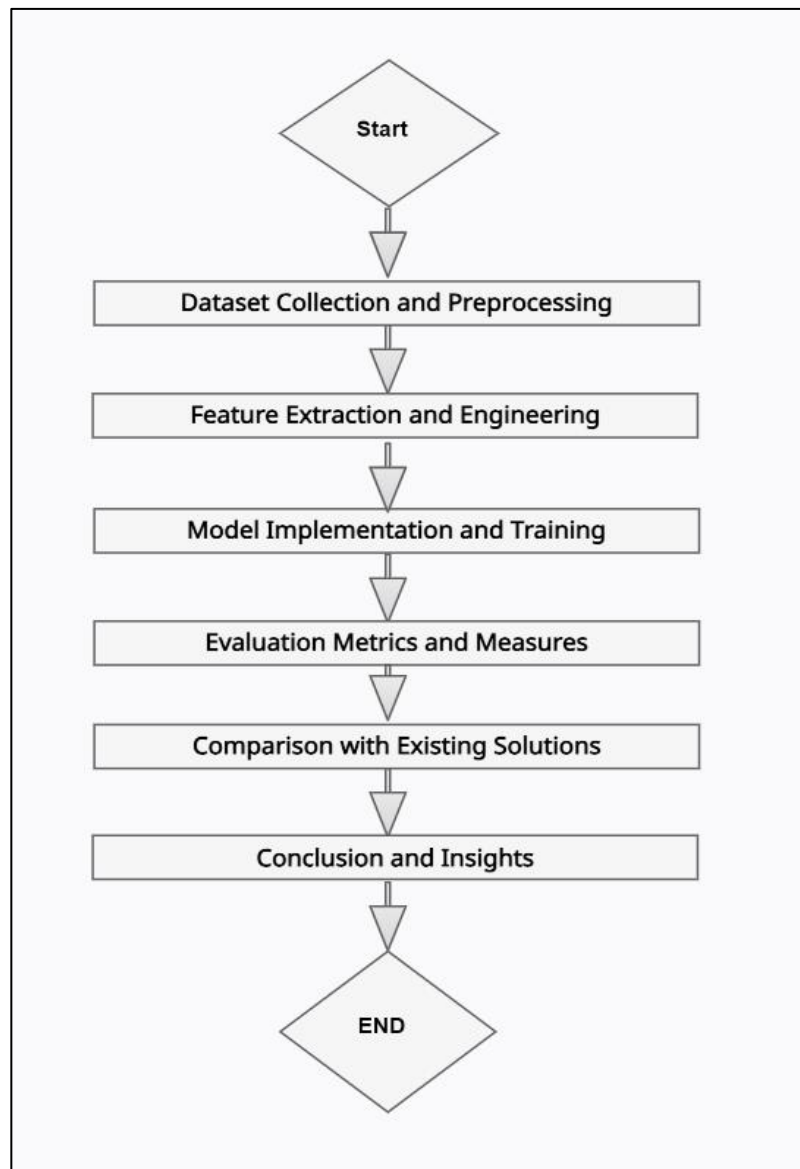


Figure 3.2: Research evaluation process flow chart

3.4 Data Collection

The data collection process for this research is meticulously designed to ensure the acquisition of comprehensive and diverse datasets, incorporating fresh data from a well-known dataset acknowledged in the research community.

3.4.1 Integration of Well-known Datasets

Recognizing the value of existing datasets well-known in the research community, we carefully selected and integrated updated datasets into our research framework. The CICIoT2023 dataset, discovered using a real-time network and benchmark for large-scale attacks in the IoT environment (Neto, E.C. et al., 2023), has been preprocessed and stored in a unified format to ensure seamless integration. This dataset augmentation serves multiple purposes; including validating our machine-learning models against known network anomalies and assessing generalization capabilities across diverse datasets. Datasets like CICIoT2023 serve as benchmarks for evaluating the performance of our anomaly detection models and facilitate comparisons with established studies (Neto, E.C. et al., 2023).

Developed by the Canadian Institute for Cybersecurity, the CICIoT2023 dataset is designed for evaluating machine learning algorithms in detecting network anomalies within IoT environments. It contains data on 33 types of attacks categorized into seven groups: DDoS, DoS, Reconnaissance, Web-based attacks, Brute force, Spoofing, and Mirai, making it a comprehensive resource for large-scale attack analysis and security research. This dataset is 13GB in size and includes approximately 47 million records (Neto, E.C. et al., 2023).

4. CHAPTER - EVALUATION AND RESULTS

In this section, we assess the effectiveness of the trained network anomaly detection model across various machine learning algorithms. The evaluation aims to provide insights into the comparative performance of different algorithms in the context of detecting anomalies within network traffic.

4.1 Training the Model

In this project, the model was trained using various machine learning methods. Figure 4.1 shows the trained model program implemented in Python using the 'scikit-learn' ML development library.

```
# Load the dataset
print("Loading dataset...")
data = pd.read_csv("data2.csv")
X = data.drop("label", axis=1)
y = data["label"]

# Split the data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y) # Stratified sampling

# Create and fit the StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

# Save the trained scaler
scaler_filename = "scaler.joblib"
joblib.dump(scaler, scaler_filename)
print(f"Trained scaler saved as '{scaler_filename}'")

# Create and fit the LabelEncoder
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)

# Save the trained LabelEncoder
label_encoder_filename = "label_encoder.joblib"
joblib.dump(label_encoder, label_encoder_filename)
print(f"LabelEncoder saved as '{label_encoder_filename}'")

# Define and train the model
print("Creating and training the Random Forest model...")
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train_scaled, y_train_encoded)

# Save the trained model
model_filename = "random_forest_model.joblib"
joblib.dump(random_forest_model, model_filename)
print(f"Trained model saved as '{model_filename}'")
```

Figure 4.1: Program used for the trained model generation

The initial step involved loading the dataset from a CSV file named "data2.csv" using the 'pandas' library. Subsequently, the dataset was split into training and validation sets using the 'train_test_split' function from scikit-learn, with a focus on maintaining a balanced distribution of target classes through stratified sampling. To ensure effective model training, I performed essential data preprocessing steps. Standard scaling was applied to the training set

using the 'StandardScaler' from scikit-learn, standardizing features to have a mean of 0 and a standard deviation of 1. Simultaneously, label encoding was employed on the target variable (y) using the 'LabelEncoder' to convert categorical labels into numerical format.

For the actual model training, I opted for a Random Forest classifier, leveraging the 'RandomForestClassifier' from scikit-learn. This choice was motivated by the model's ability to handle diverse datasets and its robustness against overfitting. I trained the model on the scaled training data and saved both the trained scaler and label encoder for potential future use. The evaluation of the model's performance was a crucial aspect of this project. Predictions were made on the validation set, and the accuracy of the model was computed using the 'accuracy_score' function. Furthermore, a detailed classification report was generated using the 'classification_report' function, providing insights into precision, recall, and F1-score for each class as shown in Figure 4.2.

```
# Make predictions on the validation set
print("Making predictions on the validation set...")
X_val_scaled = scaler.transform(X_val)
y_pred_val = random_forest_model.predict(X_val_scaled)

# Decode the predictions using the LabelEncoder
y_pred_val_decoded = label_encoder.inverse_transform(y_pred_val)

# Evaluate model performance on the validation set
print("Evaluating model performance on the validation set...")
accuracy_val = accuracy_score(y_val, y_pred_val_decoded)
print("Validation Accuracy:", accuracy_val)

# Print the classification report
print("Classification Report on Validation Set:")
print(classification_report(y_val, y_pred_val_decoded, zero_division=1))
```

Figure 4.2: Program used for the trained model performance classification

To enhance scalability and facilitate future use, it has employed the 'joblib' library to save the trained scaler, label encoder, and Random Forest model. This not only streamlines the deployment process but also ensures reusability of the trained components. In reflection, the project provided valuable insights into the application of a Random Forest model for classification tasks. The comprehensive evaluation metrics, including accuracy and the classification report, shed light on the model's strengths and potential areas for improvement. It is important to evaluate the dataset we are using in the above step. Based on the nature of the dataset, it should decide the ML algorithm that goanna be most effective for our anomalies detection process.

Evaluating the datasheet further, it has listed the various features that we used for traffic pattern analysis have listed in figure 4.3 with the descriptions (Neto, E.C. et al., 2023).

#	Feature	Description
1	ts	Timestamp
2	flow duration	Duration of the packet's flow
3	Header Length	Header Length
4	Protocol Type	IP, UDP, TCP, IGMP, ICMP, Unknown (Integers)
5	Duration	Time-to-Live (ttl)
6	Rate	Rate of packet transmission in a flow
7	Srate	Rate of outbound packets transmission in a flow
8	Drate,	Rate of inbound packets transmission in a flow
9	fin flag number	Fin flag value
10	syn flag number	Syn flag value
11	rst flag number	Rst flag value
12	psh flag numbe	Psh flag value
13	ack flag number	Ack flag value
14	ece flag numbe	Ece flag value
15	cwr flag number	Cwr flag value
16	ack count	Number of packets with ack flag set in the same flow
17	syn count	Number of packets with syn flag set in the same flow
18	fin count	Number of packets with fin flag set in the same flow
19	urg coun	Number of packets with urg flag set in the same flow
20	rst count	Number of packets with rst flag set in the same flow
21	HTTP	Indicates if the application layer protocol is HTTP
22	HTTPS	Indicates if the application layer protocol is HTTPS
23	DNS	Indicates if the application layer protocol is DNS
24	Telnet	Indicates if the application layer protocol is Telnet
25	SMTP	Indicates if the application layer protocol is SMTP
26	SSH	Indicates if the application layer protocol is SSH
27	IRC	Indicates if the application layer protocol is IRC
28	TCP	Indicates if the transport layer protocol is TCP
29	UDP	Indicates if the transport layer protocol is UDP
30	DHCP	Indicates if the application layer protocol is DHCP
31	ARP	Indicates if the link layer protocol is ARP
32	ICMP	Indicates if the network layer protocol is ICMP
33	IPv	Indicates if the network layer protocol is IP
34	LLC	Indicates if the link layer protocol is LLC
35	Tot sum	Summation of packets lengths in flow
36	Min	Minimum packet length in the flow
37	Max	Maximum packet length in the flow
38	AVG	Average packet length in the flow
39	Std	Standard deviation of packet length in the flow
40	Tot size	Packet's length
41	IAT	The time difference with the previous packet
42	Number	The number of packets in the flow
43	Magnitude	(Average of the lengths of incoming packets in the flow + average of the lengths of outgoing packets in the flow) ^{0.5}
44	Radius	(Variance of the lengths of incoming packets in the flow + variance of the lengths of outgoing packets in the flow) ^{0.5}
45	Covariance	Covariance of the lengths of incoming and outgoing packets
46	Variance	Variance of the lengths of incoming packets in the flow / variance of the lengths of outgoing packets in the flow
47	Weight	Number of incoming packets × Number of outgoing packets

Figure 4.3: Dataset Features with descriptions (Neto, E.C. et al., 2023)

When training the model, it was using two approaches. One is based on the 34 classes and the other one based on the binary classes. Here, 'BenignTraffic' refers to normal traffic and other belongs to anomalies.

ACK fragmentation, UDP flood, SlowLoris, ICMP flood, RSTFIN flood, PSHACK flood, HTTP flood, UDP fragmentation, TCP flood, SYN flood, SynonymousIP flood, Dictionary brute force, Arp spoofing, DNS spoofing, TCP flood, HTTP flood, SYN flood, UDP flood, Ping sweep, OS scan, Vulnerability scan, Port scan, Host discovery, Ping sweep, OS scan, Vulnerability scan, Port scan, Host discovery, GREIP flood, Greeth flood, and UDPPlain were among the 33 types of attacks included in the dataset (Neto, E.C. et al., 2023).

Knowing that other data types except 'BenignTraffic' refers to anomalies; it has created the binary classes for the same dataset by updating 'BenignTraffic' labels as "Normal" and other 33 classes as the "Anomaly". Based on that it would be allowing us for both multi class and binary class based trained models.

Similarly, we can execute the same trained model program with processed data that has binary classes and Figure 4.4 showing trained model program output belonged to the binary classes.

```
import pandas as pd
Loading dataset...
Trained scaler saved as 'scaler.joblib'
LabelEncoder saved as 'label_encoder.joblib'
Creating and training the Random Forest model...
Trained model saved as 'random_forest_model.joblib'
Making predictions on the validation set...
Evaluating model performance on the validation set...
Validation Accuracy: 0.9965645816749759
Classification Report on Validation Set:
```

	precision	recall	f1-score	support
Anomaly	1.00	1.00	1.00	46618
Normal	0.92	0.94	0.93	1120
accuracy			1.00	47738
macro avg	0.96	0.97	0.96	47738
weighted avg	1.00	1.00	1.00	47738

Figure 4.4: Training model program output with binary classes

Based on that 2 set of classes, we can input two different datasets for the training model program. Hence, Figure 4.5 has showing the training model program output with multiple classes.

```

import pandas as pd
Loading dataset...
Trained scaler saved as 'scaler.joblib'
LabelEncoder saved as 'label_encoder.joblib'
Creating and training the Random Forest model...
Trained model saved as 'random_forest_model.joblib'
Making predictions on the validation set...
Evaluating model performance on the validation set...
Validation Accuracy: 0.9912229251330177
Classification Report on Validation Set:

```

	precision	recall	f1-score	support
Backdoor_Malware	1.00	0.00	0.00	4
BenignTraffic	0.84	0.98	0.90	1120
BrowserHijacking	1.00	0.00	0.00	6
CommandInjection	1.00	0.17	0.29	6
DDoS-ACK_Fragmentation	0.99	0.99	0.99	301
DDoS-HTTP_Flood	0.94	0.88	0.91	34
DDoS-ICMP_Flood	1.00	1.00	1.00	7311
DDoS-ICMP_Fragmentation	0.99	0.99	0.99	475
DDoS-PSHACK_Flood	1.00	1.00	1.00	4242
DDoS-RSTFINFlood	1.00	1.00	1.00	4134
DDoS-SYN_Flood	1.00	1.00	1.00	4148
DDoS-SlowLoris	0.86	0.90	0.88	21
DDoS-SynonymousIP_Flood	1.00	1.00	1.00	3638
DDoS-TCP_Flood	1.00	1.00	1.00	4630
DDoS-UDP_Flood	1.00	1.00	1.00	5525
DDoS-UDP_Fragmentation	0.99	0.99	0.99	297
DNS_Spoofing	0.73	0.62	0.67	185
DictionaryBruteForce	1.00	0.08	0.14	13
DoS-HTTP_Flood	0.99	0.98	0.98	83
DoS-SYN_Flood	1.00	1.00	1.00	2055
DoS-TCP_Flood	1.00	1.00	1.00	2726
DoS-UDP_Flood	1.00	1.00	1.00	3391
MITM-ArpSpoofing	0.85	0.79	0.82	323
Mirai-greeth_flood	1.00	0.99	1.00	1003
Mirai-greip_flood	1.00	0.99	0.99	752
Mirai-udpplain	1.00	1.00	1.00	932
Recon-HostDiscovery	0.78	0.78	0.78	139
Recon-OSScan	0.71	0.33	0.45	103
Recon-PingSweep	1.00	0.00	0.00	1
Recon-PortScan	0.80	0.56	0.66	86
SqlInjection	1.00	0.00	0.00	6
Uploading_Attack	1.00	0.00	0.00	2
VulnerabilityScan	0.80	0.93	0.86	42
XSS	1.00	0.00	0.00	4
accuracy			0.99	47738
macro avg	0.95	0.70	0.71	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.5: Training model program output with multiple classes

If we closely review the output of the trained model in contrast view for both multiple and binary classes, we can see that binary classes model performing much better as we composed the outputs in the both programs and shown in the Table 4.1.

Random Forest ML algorithm	Validation accuracy	precision	recall	f1-score
34 classes	0.991222925	0.99	0.99	0.99
2 classes	0.996564582	1	1	1

Table 4.1: Model performance in Random forest algorithm with binary and multiple classes

As per the result we obtained, it can be said that there would be improved performances with binary class implementation in anomalies detection trained model. More importantly, it is showing great validation accuracy in terms of anomalies detection.

4.2 Testing the Trained Data Model

In the testing phase of this project, I extended the machine learning pipeline developed during the training phase to assess the model's performance on an external dataset. The test dataset was loaded from a CSV file named "test_data2.csv" using the 'pandas' library, mirroring the initial data loading step in the training phase. In Figure 4.6, it has shown the test dataset that we are using for test our network anomalies detection model that trained using a ML algorithm.

AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT
Tot sum	Min	Max	AVG	Std	Tot size	IAT	Number	Magnitue	Radius	Covarianc	Variance	Weight
357	62	66	65.37778	1.428539	62	0.005144	5.5	11.43426	2.02026	10.35062	0.2	38.5
441	42	42	42	0	42	83128166	9.5	9.165151	0	0	0	141.55
567	54	54	54	0	54	83089077	9.5	10.3923	0	0	0	141.55
567	54	54	54	0	54	82946739	9.5	10.3923	0	0	0	141.55
525	50	50	50	0	50	83123872	9.5	10	0	0	0	141.55
1665.12	143.6	160.16	156.3002	6.455014	160.16	83008166	9.5	17.65033	9.137735	238.2624	0.18	141.55
567	54	54	54	0	54	82955901	9.5	10.3923	0	0	0	141.55
441	42	42	42	0	42	83149447	9.5	9.165151	0	0	0	141.55
441	42	42	42	0	42	83483283	9.5	9.165151	0	0	0	141.55
1450.4	43	186.7	93.31802	46.85288	101.2	1.67E+08	13.5	13.65272	66.42284	2215.252	1	244.6

Figure 4.6: Test dataset

To maintain consistency with the preprocessing steps applied during training, I checked for the presence of the 'label' column in the test data. If present, true labels were extracted for later comparison, and the 'label' column was removed from the features. In cases where the 'label' column was absent, a placeholder for true labels ('Unknown') was created, ensuring a seamless transition between datasets.

To ensure uniform preprocessing, the same scaler and label encoder used during training were loaded. The StandardScaler from scikit-learn was employed to scale numerical features, aligning with the preprocessing methodology applied to the training data. The LabelEncoder converted categorical labels into numerical format, maintaining the encoding consistency established during model training.

The trained Random Forest model, previously saved as "random_forest_model.joblib," was loaded to make predictions on the scaled test data. Predictions were decoded using the label encoder, restoring them to their original categorical form for a meaningful comparison with the true labels.

A comparison DataFrame, named 'results_df,' was generated to juxtapose the true and predicted labels. This facilitated a detailed inspection of the model's performance on the external dataset, to the evaluation process conducted during training.

The utilization of the joblib library was extended to load the necessary components, including the scaler and label encoder, ensuring reproducibility and consistency between the training and testing phases. This approach not only streamlined the testing process but also demonstrated the scalability and reusability of the machine learning pipeline.

In reflection, the testing phase provided valuable insights into the model's generalization capabilities on unseen data. The comparison DataFrame, akin to the classification report during training, offered a comprehensive view of the model's accuracy and potential areas for improvement.

In Figure 4.7, it shows the testing program used to test the anomaly detection model.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
import joblib

# Load the test dataset
print("Loading test dataset...")
test_data = pd.read_csv("test_data2.csv")

# Check if 'label' column exists in the test data
if 'label' in test_data.columns:
    # Extract the 'label' column for later comparison
    true_labels = test_data['label']

    # Drop the 'label' column
    test_data = test_data.drop(['label'], axis=1)
else:
    # If 'label' column does not exist, create a placeholder for true labels
    true_labels = pd.Series(['Unknown'] * len(test_data))

# Load the scaler and label encoder used during training
print("Loading scaler and label encoder used during training...")
scaler = joblib.load("scaler.joblib")
label_encoder = joblib.load("label_encoder.joblib")

# Scale numerical features
print("Scaling numerical features...")
X_test_scaled = scaler.transform(test_data)

# Load the trained Random Forest model from the saved file
print("Loading trained Random Forest model...")
model_filename = "random_forest_model.joblib"
trained_model = joblib.load(model_filename)

# Make predictions on the scaled test data
print("Making predictions on the test data...")
predictions = trained_model.predict(X_test_scaled)

# Decode predictions using the label encoder
decoded_predictions = label_encoder.inverse_transform(predictions)

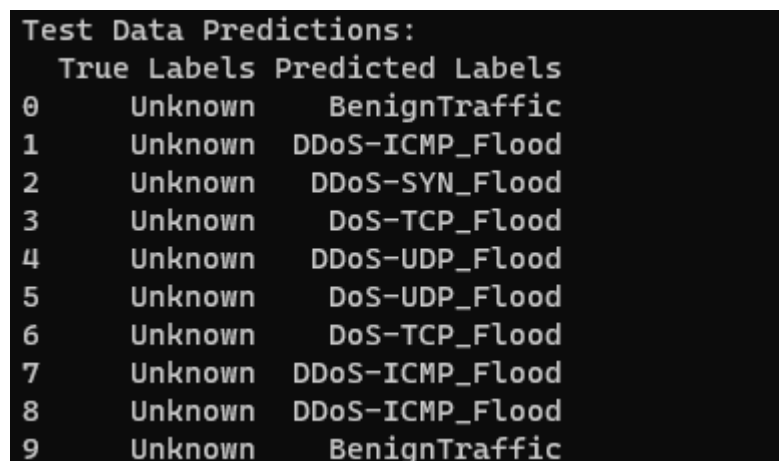
# Create a DataFrame to compare true labels with predicted labels
results_df = pd.DataFrame({'True Labels': true_labels, 'Predicted Labels': decoded_predictions})

# Output the DataFrame to inspect the results
print("Test Data Predictions:")
print(results_df)
```

Figure 4.7: Testing program for the experiment data

It is able to generate trained models with great accuracy. In this example it has used the “Random forest” supervised learning algorithm. We will be evaluating and showing the results for the ML models in the next section. Here, it is using the test dataset shown in Figure 4.6. In this data we know what the behavior of the data is. Hence, it would be using this test data to test and evaluate the accuracy of the anomalies detection process. As discussed earlier,

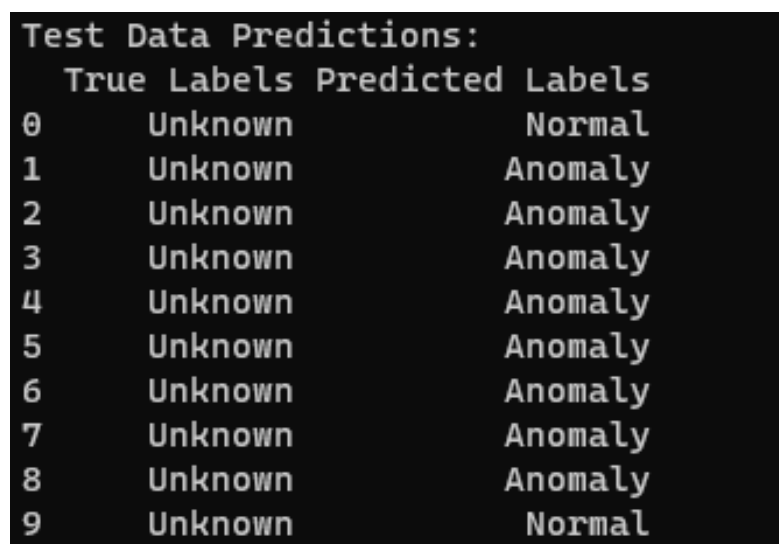
we have two ways of training the model in one algorithm, which is using a multiclass dataset and binary class data set.



Test Data Predictions:		
	True Labels	Predicted Labels
0	Unknown	BenignTraffic
1	Unknown	DDoS-ICMP_Flood
2	Unknown	DDoS-SYN_Flood
3	Unknown	DoS-TCP_Flood
4	Unknown	DDoS-UDP_Flood
5	Unknown	DoS-UDP_Flood
6	Unknown	DoS-TCP_Flood
7	Unknown	DDoS-ICMP_Flood
8	Unknown	DDoS-ICMP_Flood
9	Unknown	BenignTraffic

Figure 4.8: Multiclass dataset used trained model test outcome

Figure 4.8 has shown test results for the trained model using the multi class labeled data set. While Figure 4.9 showing the test results for the trained model using the binary class labeled data set. In both cases, it has shown 100% accuracy for the detection of those data with the trained algorithm “Random Forest”. However, we cannot expect the same outcome for all the ML algorithms and we will be evaluating different algorithms in the next section.



Test Data Predictions:		
	True Labels	Predicted Labels
0	Unknown	Normal
1	Unknown	Anomaly
2	Unknown	Anomaly
3	Unknown	Anomaly
4	Unknown	Anomaly
5	Unknown	Anomaly
6	Unknown	Anomaly
7	Unknown	Anomaly
8	Unknown	Anomaly
9	Unknown	Normal

Figure 4.9: Binary class dataset used trained model test outcome

In fact, we achieved 99.66% of accuracy and as a result were able to figure out all the test data which were provided. However, the dynamic nature of the network and security threats would encourage us for further experiments.

4.3 Evaluation of trained model performances in different ML Algorithms

In this section, we assess the effectiveness of the trained network anomaly detection model across various machine learning algorithms. The evaluation aims to provide insights into the comparative performance of different algorithms in the context of detecting anomalies within network traffic.

4.3.1 Algorithms Selection and Considerations

For the evaluation, we considered a diverse set of machine learning algorithms known for their applicability in anomaly detection. The selected algorithms encompass both supervised and unsupervised learning paradigms. We already know that we have a supervised dataset that could lead to higher performance when compared to unsupervised methods. However, we cannot neglect the unsupervised methods due to the capabilities of identifying traffic patterns without having the labeled data. On the other hand, according to the dynamic nature of networks and threats we have to consider both methodologies.

To gauge the performance of each algorithm, we employed a comprehensive set of evaluation metrics, including accuracy, precision, recall, F1-score, and area under the Receiver Operating Characteristic (ROC) curve. Other than that unsupervised learning method specific measures like Silhouette Score and Davies-Bouldin Index have been used. These metrics provide a holistic view of the model's ability to correctly classify normal and anomalous network behavior.

The trained model was evaluated using the same test dataset across all selected algorithms, ensuring a fair comparison. Results were analyzed in terms of both overall performance and algorithm-specific nuances. Key aspects under consideration included computational efficiency, scalability, and the ability to handle varying types of anomalies commonly encountered in network traffic.

To be more precise, it is using only binary classes for the evaluation since some of the algorithms are specifically built for the binary classification. Other than that, accuracy is higher for the detection models we trained for the binary classes when compared to the multiple classes.

4.3.2 Unsupervised learning algorithms

This section evaluates a few unsupervised algorithms using the same data set.

Isolation Forests

As shown in Figure 4.10 it has trained the isolation model to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the Isolation Forest model
print("Creating and training the Isolation Forest model...")
isolation_forest_model = IsolationForest(contamination=0.05, random_state=42) # Adjust contamination as needed
isolation_forest_model.fit(X_train_scaled)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = isolation_forest_model.predict(X_val_scaled)
y_pred_val = [1 if pred == -1 else 0 for pred in y_pred_val] # Convert -1 (anomaly) to 1, 0 for normal
```

Figure 4.10: Isolation forest algorithm based trained model function

In the isolation forest algorithm execution, it has received the Validation Accuracy of 0.97333. Figure 4.11 shows the classification report for the isolation algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	0.97	0.99	46618
Normal	0.47	0.97	0.63	1120
accuracy			0.97	47738
macro avg	0.73	0.97	0.81	47738
weighted avg	0.99	0.97	0.98	47738

Figure 4.11: Classification report for the isolation forest algorithm

According to those data shown in Figure 4.11, it does not have a greater accuracy when picking the normal data even though it has picked anomaly data with higher accuracy based on the higher support data.

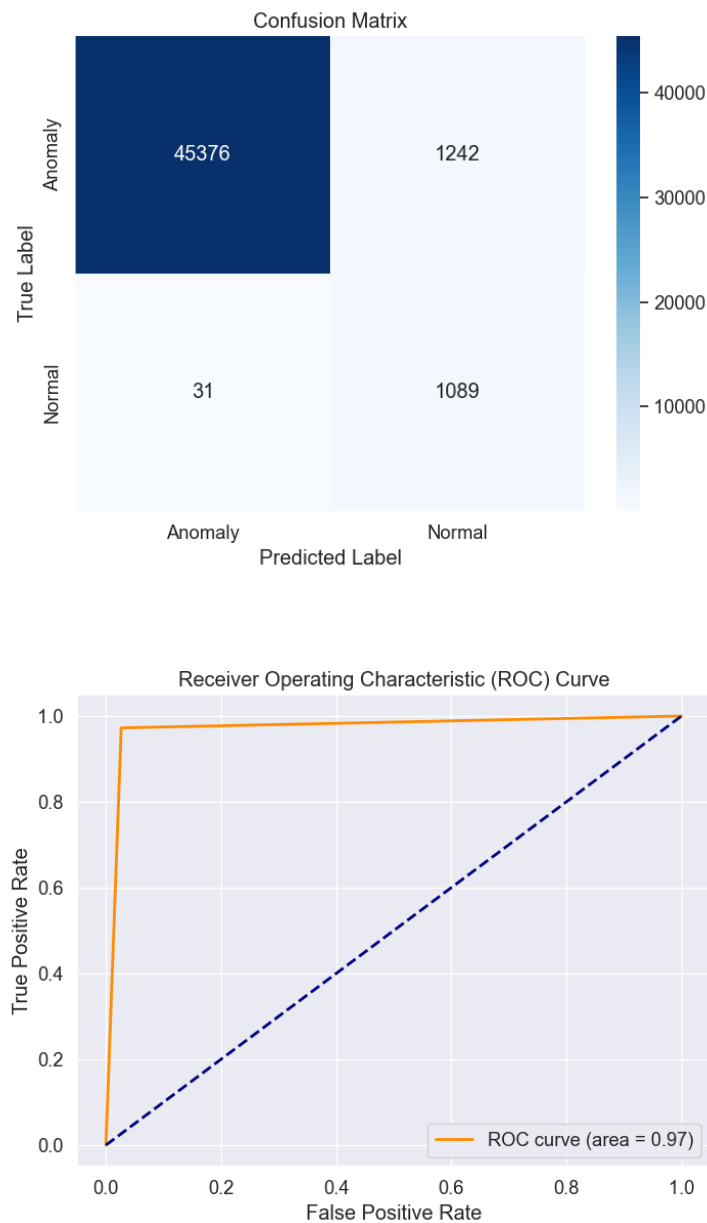


Figure 4.12: Confusion matrix and ROC Curve for isolation forest algorithm

As per confusion matrix shown in Figure 4.12, a significant amount of anomaly traffic has flagged as normal traffic by value it is 1242. In the ROC curve it has the value of 0.97 which is a good value in terms of anomalies detection.

Local Outlier Factor (LOF)

As shown in the Figure 4.13 it has trained the LOF model to evaluate its performance in the same network traffic patterns dataset.

```
# Scale numerical features
print("Standard scaling numerical features (optional)...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Define and train the Local Outlier Factor model
print("Creating and training the Local Outlier Factor model...")
lof_model = LocalOutlierFactor(n_neighbors=20, contamination=0.05) # Adjust parameters as needed
y_scores = -lof_model.fit_predict(X_val_scaled) # Negative scores for compatibility with roc_curve

# Convert LOF scores to binary predictions (1 for anomaly, 0 for normal)
y_pred_val = [1 if score > 0 else 0 for score in y_scores]
```

Figure 4.13: LOF algorithm based trained model function

In the LOF algorithm execution, it has received the Validation Accuracy of 0.92679. Figure 4.14 shows the classification report for the LOF algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	0.98	0.95	0.96	46618
Normal	0.00	0.01	0.00	1120
accuracy			0.93	47738
macro avg	0.49	0.48	0.48	47738
weighted avg	0.95	0.93	0.94	47738

Figure 4.14: Classification report for the LOF algorithm

According to those data shown in Figure 4.14, it does not have a greater accuracy when picking the normal data even though it has picked anomaly data with higher accuracy based on the higher support data.

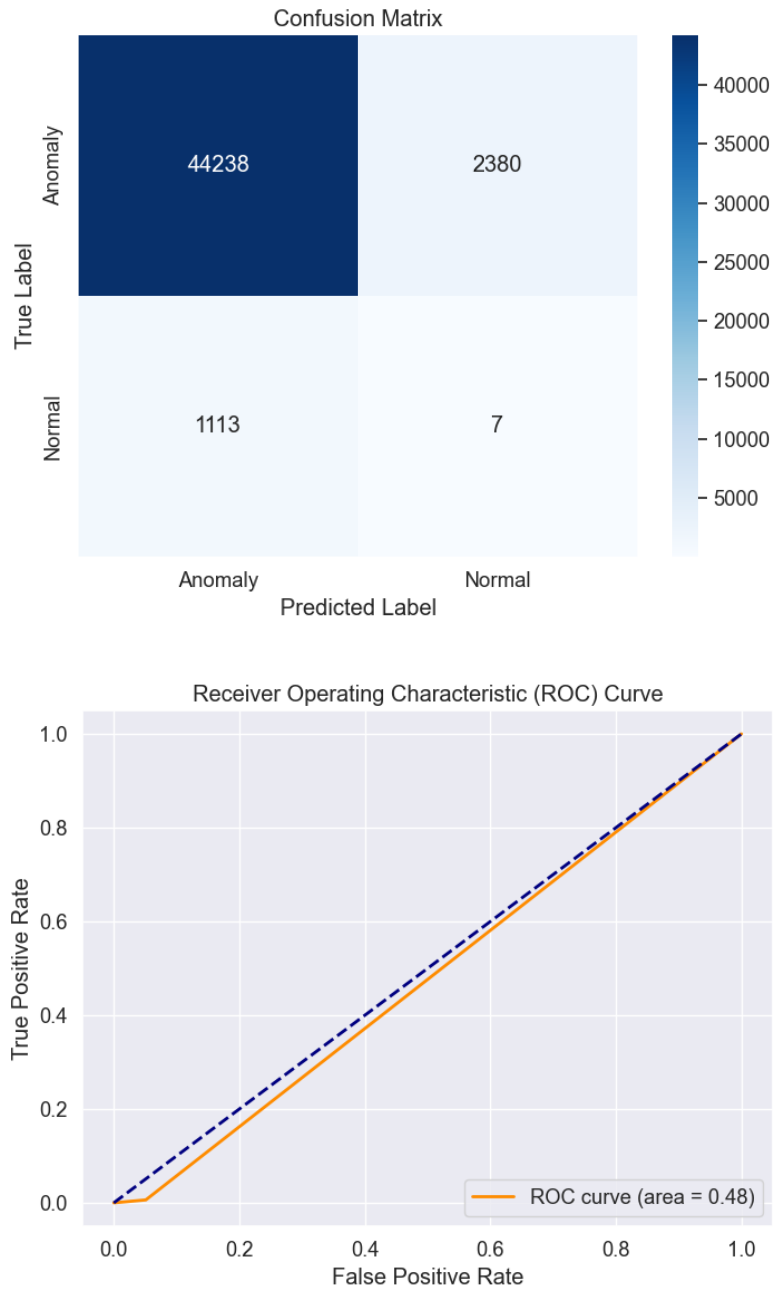


Figure 4.15: Confusion matrix and ROC Curve for LOF algorithm

As per confusion matrix shown in Figure 4.15, a significant amount of anomaly traffic has flagged as normal traffic by value it is 2380 and normal traffic of 1113 predicted as Anomaly. In the ROC curve it has the value of 0.48 which is a very poor value in terms of anomalies detection as it shows poor ability of distinguishing normal traffic and anomaly traffic.

Autoencoders

As shown in Figure 4.16 it has trained the detection model using Autoencoders algorithm to evaluate its performance of the model in the same network traffic patterns dataset.

```
# Scale numerical features
print("Standard scaling numerical features (optional)...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Define and train the Local Outlier Factor model
print("Creating and training the Local Outlier Factor model...")
lof_model = LocalOutlierFactor(n_neighbors=20, contamination=0.05) # Adjust parameters as needed
y_scores = -lof_model.fit_predict(X_val_scaled) # Negative scores for compatibility with roc_curve

# Convert LOF scores to binary predictions (1 for anomaly, 0 for normal)
y_pred_val = [1 if score > 0 else 0 for score in y_scores]
```

Figure 4.16: Autoencoders algorithm based trained model function

In the Autoencoders algorithm execution, it has received the Validation Accuracy of 0.96856.

Figure 4.17 shows the classification report for the Autoencoders algorithm execution.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	0.97	0.98	46618
Normal	0.42	0.90	0.57	1120
accuracy			0.97	47738
macro avg	0.71	0.93	0.78	47738
weighted avg	0.98	0.97	0.97	47738

Figure 4.17: Classification report for the Autoencoders algorithm

According to those data shown in Figure 4.17, it does not have a greater accuracy when picking the normal data even though it has picked anomaly data with higher accuracy based on the higher support data.

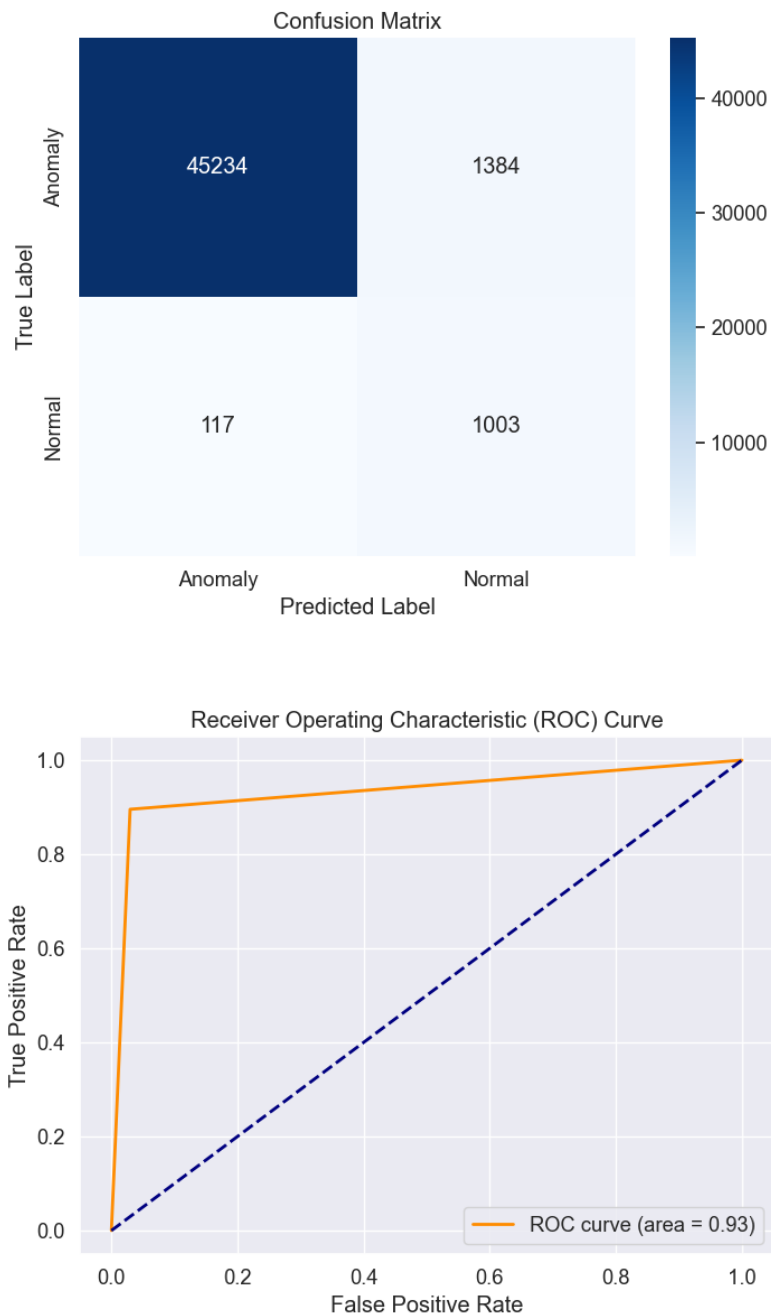


Figure 4.18: Confusion matrix and ROC Curve for Autoencoders algorithm

As per confusion matrix shown in Figure 4.18, a significant amount of anomaly traffic has flagged as normal traffic. In the ROC curve it has the value of 0.93 and in comparison that is not a most impressive score. In that case, we have to evaluate it as an average one since that data sample used has less amount of normal traffic, it can be assumed that it might perform well with the higher support of data.

4.3.3 Supervised learning algorithms

In this section it has evaluated a few supervised learning algorithms using the same data set.

Random Forests

As shown in the Figure 4.19 it has trained the detection model using the Random Forests algorithm to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the Random Forest model
print("Creating and training the Random Forest model...")
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = random_forest_model.predict(X_val)
y_prob_val = random_forest_model.predict_proba(X_val)
```

Figure 4.19: Random forest algorithm based trained model function

In the Random Forests algorithm execution, it has received the Validation Accuracy of 0.99642. Figure 4.20 shows the classification report for the Random Forests algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	1.00	1.00	46618
Normal	0.92	0.93	0.92	1120
accuracy			1.00	47738
macro avg	0.96	0.96	0.96	47738
weighted avg	1.00	1.00	1.00	47738

Figure 4.20: Classification report for the random forest algorithm

According to those data shown in Figure 4.20, it has great accuracy when picking both normal data and anomaly data.

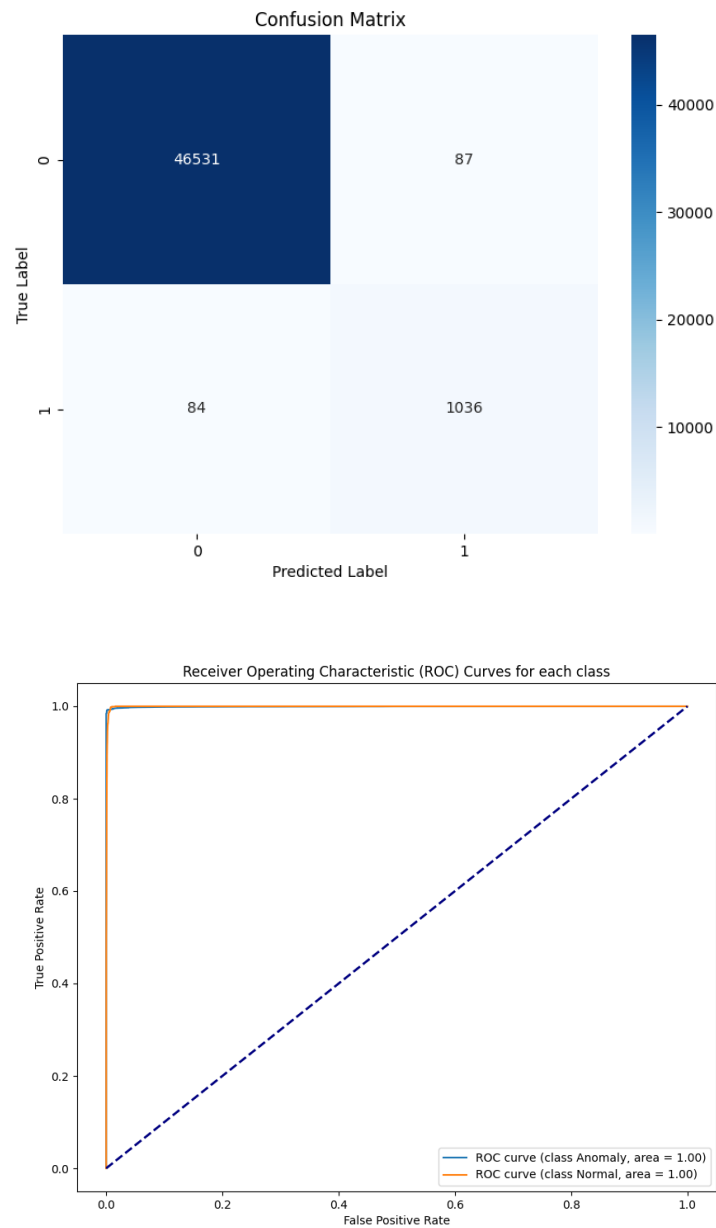


Figure 4.21: Confusion matrix and ROC Curve for random forest algorithm

As per confusion matrix shown in the Figure 4.21, both anomaly data and normal data has picked with a greater accuracy providing that around 0.002% of the traffic was not picked accurately. In the ROC curve it has the value of 1.000 for the AUC in both cases.

Logistic Regression

As shown in Figure 4.22 it has trained the detection model using Logistic Regression algorithm to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the model
print("Creating and training the model... (C=1 for faster training)")
model = LogisticRegression(C=1, random_state=42, max_iter=1000) # Increase max_iter

# Suppress the specific warning for precision being ill-defined
with warnings.catch_warnings():
    warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
    model.fit(X_train_red, y_train_red)
```

Figure 4.22: Logistic Regression algorithm based trained model function

In the Logistic Regression algorithm execution, it has received the Validation Accuracy of 0.98835. Figure 4.23 shows the classification report for the Logistic Regression algorithm.

Classification Report:				
	precision	recall	f1-score	support
Anomaly	0.99	1.00	0.99	46618
Normal	0.79	0.71	0.75	1120
accuracy			0.99	47738
macro avg	0.89	0.85	0.87	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.23: Classification report for the Logistic Regression algorithm

According to those data showed in Figure 4.23, it does have a greater accuracy when picking the normal data and has picked anomaly data with higher accuracy around 80% based on the support data. As per those results precision, recall and f1-score is not that great for the normal data. We can assume that lower support of the data set to the normal traffic would be a reason for this outcome.

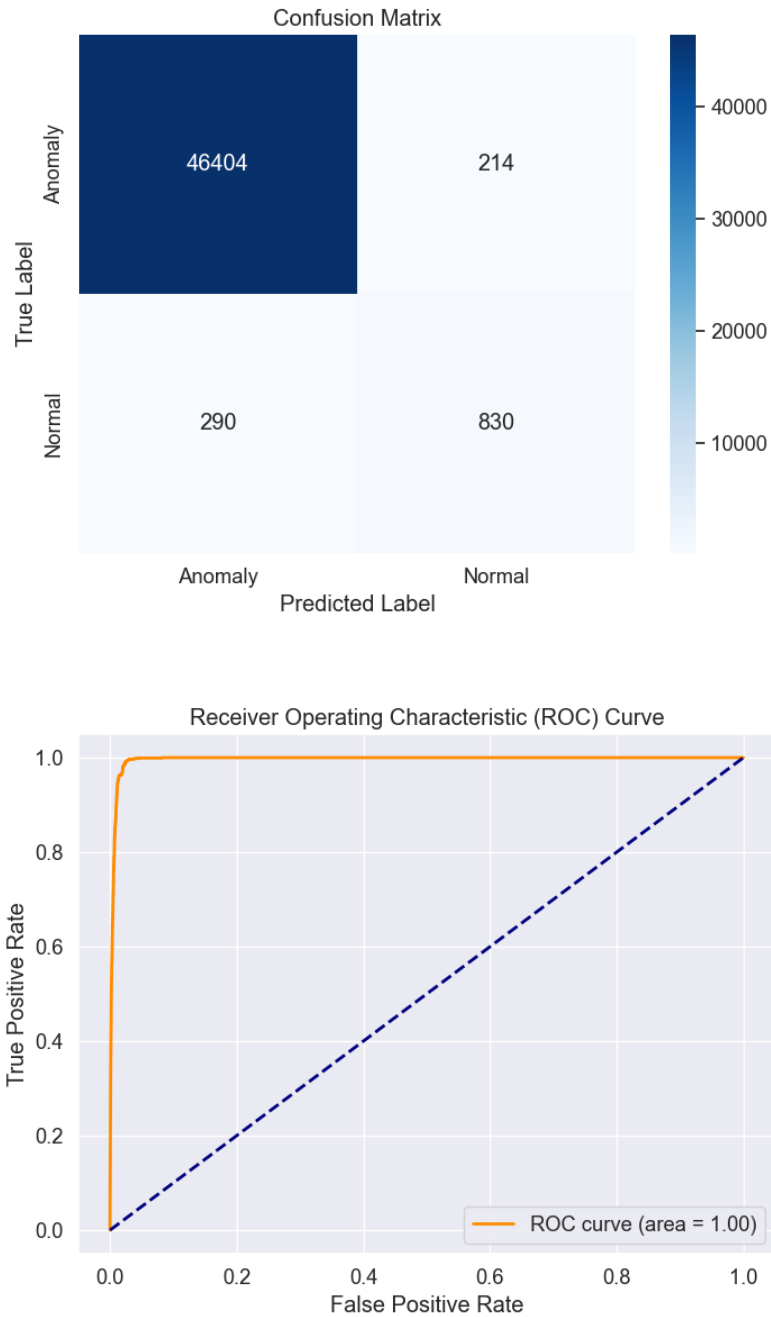


Figure 4.24: Confusion matrix and ROC Curve for Logistic Regression algorithm

As per confusion matrix shown in the Figure 4.24, a significant amount of anomaly traffic has flagged as normal traffic by value it is 214 and normal traffic of 290 predicted as Anomaly. In the ROC curve it has the value of 1.00 which is a very good value in terms of anomalies detection as it shows good ability of distinguishing normal traffic and anomaly traffic.

Decision Trees

As shown in the Figure 4.25 it has trained the model using the Decision Trees algorithm to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the Decision Tree model
print("Creating and training the Decision Tree model...")
decision_tree_model = DecisionTreeClassifier(max_depth=5, random_state=42)
decision_tree_model.fit(X_train_scaled, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = decision_tree_model.predict(X_val_scaled)
```

Figure 4.25: Decision Trees algorithm based trained model function

In the Decision Trees algorithm execution, it has received the Validation Accuracy of 0.99424.

Figure 4.26 shows the classification report for the Decision Trees algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	1.00	1.00	46618
Normal	0.86	0.90	0.88	1120
accuracy			0.99	47738
macro avg	0.93	0.95	0.94	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.26: Classification report for the Decision Trees algorithm

According to those data shown in Figure 4.26, it does not have a greater accuracy when picking the normal data and anomaly data with higher accuracy based on the higher support data.

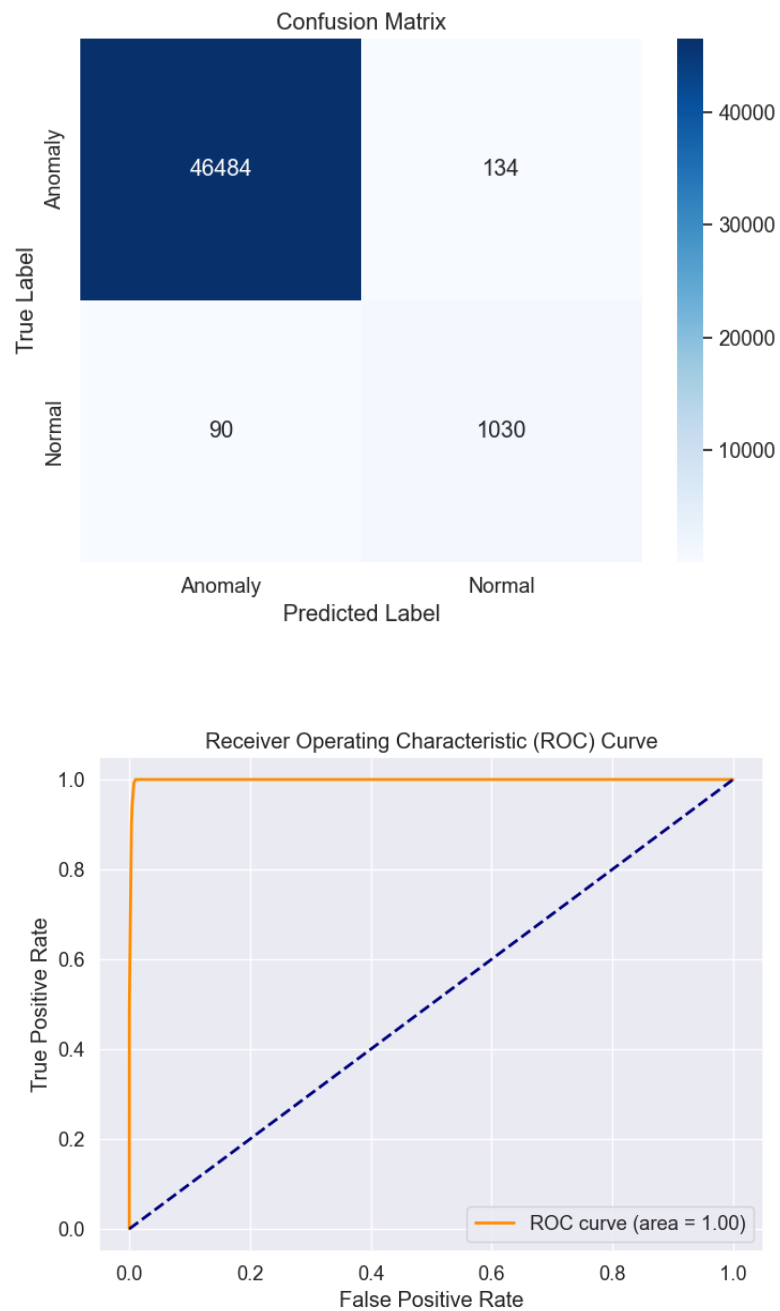


Figure 4.27: Confusion matrix and ROC Curve for Decision Trees algorithm

As per confusion matrix shown in Figure 4.27, both traffic of data recognized in accurate manner. In the ROC curve it has the value of 1.00 which is a great value in terms of anomalies detection.

K-Nearest Neighbors (KNN)

As shown in Figure 4.28 it has trained the detection model using K-Nearest Neighbors algorithm to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the knn model
print("Creating and training the knn model...")
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = knn.predict(X_val)
```

Figure 4.28: K-Nearest Neighbors algorithm based trained model function

In the K-Nearest Neighbors algorithm execution, it has received the Validation Accuracy of 0.99045. Figure 4.31 shows the classification report for the isolation algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	0.99	0.99	46618
Normal	0.74	0.82	0.77	1120
accuracy			0.99	47738
macro avg	0.87	0.90	0.88	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.29: Classification report for the K-Nearest Neighbors algorithm

According to those data shown in Figure 4.29, it does have a greater accuracy when picking both normal anomaly data. However, it has showing values 0.74, 0.82, and 0.77 respectively for the precision, recall and f1-score for the normal data detection. Still can rate this as a good one for network traffic anomalies detection due to the higher accuracy based on the higher support data.

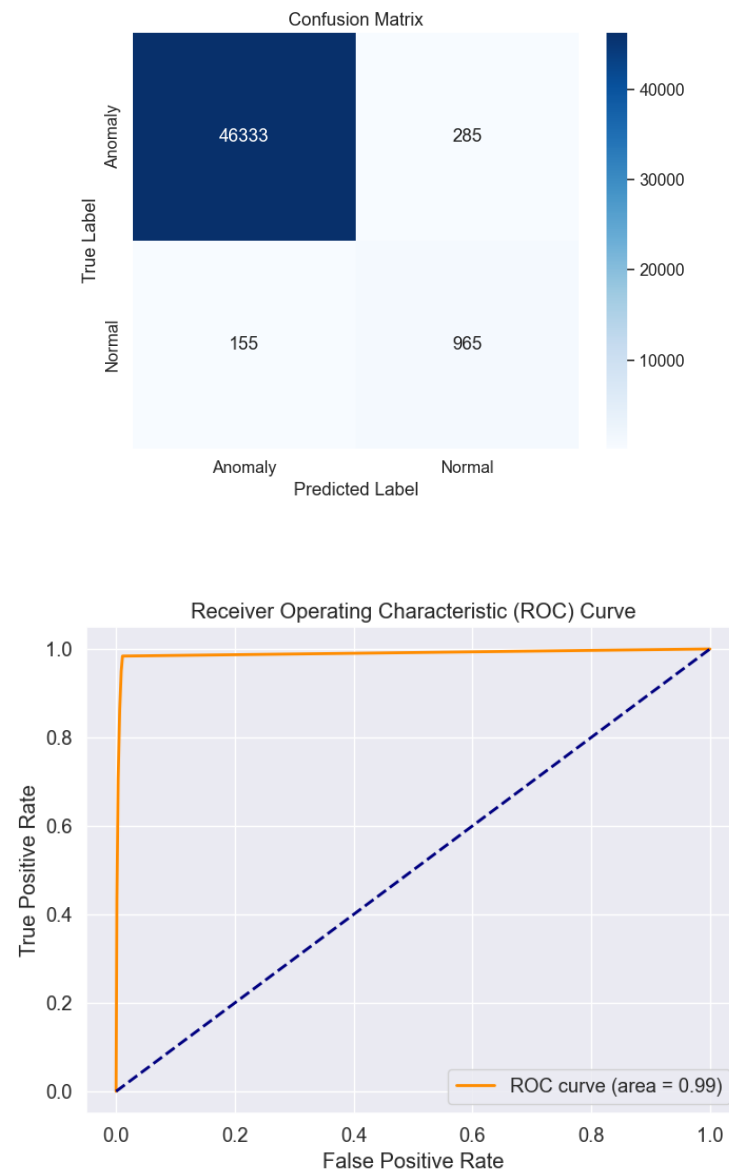


Figure 4.30: Confusion matrix and ROC Curve for K-Nearest Neighbors algorithm

As per confusion matrix shown in the Figure 4.30, a significant amount of anomaly traffic has flagged as normal traffic by value it is 285. However, with the validation accuracy of 0.99+ still can rate as a good choice for the network anomalies detection. In the ROC curve it has the value of 0.99 which is a good value in terms of anomalies detection.

Naive Bayes

As shown in Figure 4.31, it has trained the network anomaly detection model using Naive Bayes algorithm to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the Gaussian Naive Bayes model
print("Creating and training the Gaussian Naive Bayes model...")
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = naive_bayes_model.predict(X_val)
y_prob_val = naive_bayes_model.predict_proba(X_val)
```

Figure 4.31: Naive Bayes algorithm based trained model function

In the Naive Bayes algorithm execution, it has received the Validation Accuracy of 0.95479.

Figure 4.32 shows the classification report for the Naive Bayes algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	0.95	0.98	46618
Normal	0.35	1.00	0.51	1120
accuracy			0.96	47738
macro avg	0.67	0.98	0.74	47738
weighted avg	0.98	0.96	0.97	47738

Figure 4.32: Classification report for the Naive Bayes algorithm

According to those data shown in Figure 4.32, it does not have a greater accuracy when picking the normal data even though it has picked anomaly data with higher accuracy based on the higher support data.

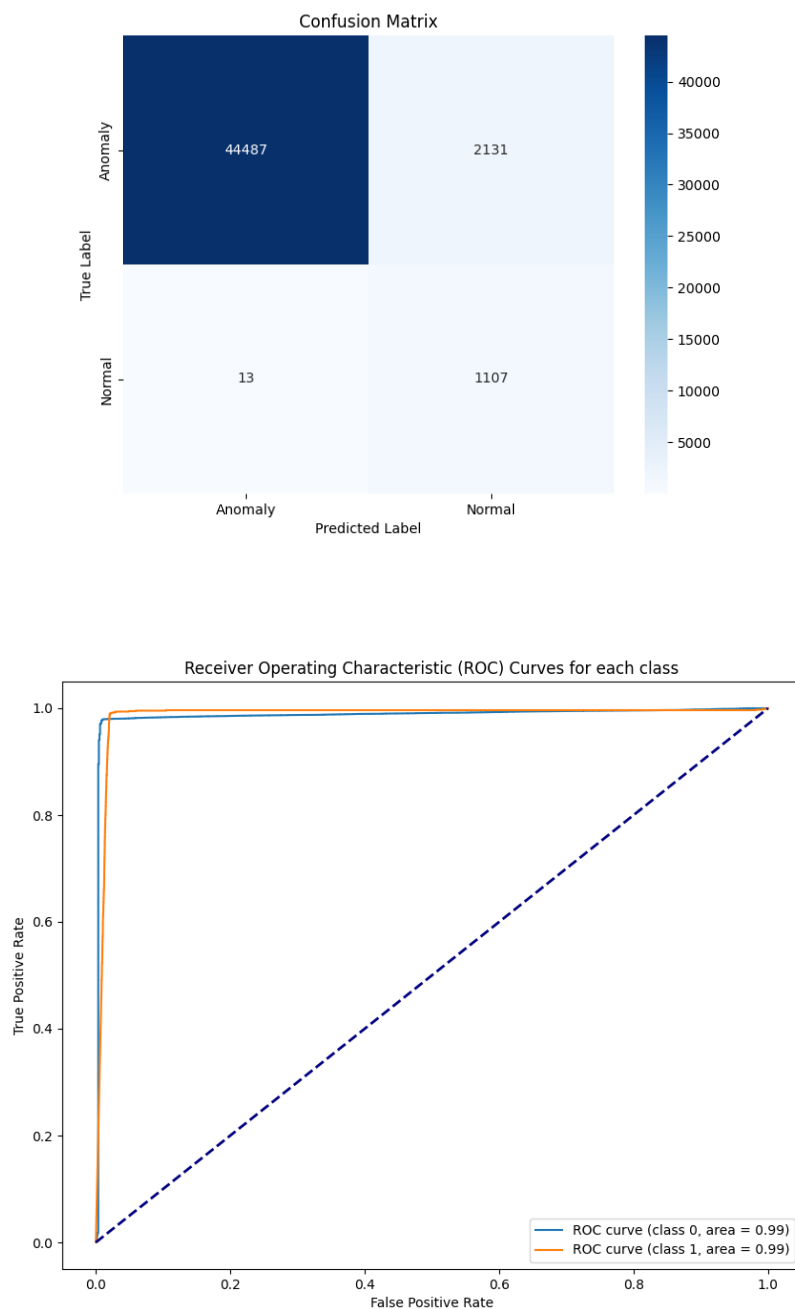


Figure 4.33: Confusion matrix and ROC Curve for Naive Bayes algorithm

As per confusion matrix shown in the Figure 4.33, a significant amount of anomaly traffic has flagged as normal traffic by value it is 2131 and except that it showed a good performance. In the ROC curve it has the value of 0.99 each for both classes which is good in terms of anomalies detection as it shows poor ability of distinguishing normal traffic and anomaly traffic.

AdaBoost (Adaptive Boosting)

As shown in Figure 4.34 it has trained the anomalies detection model using AdaBoost algorithm in order to evaluate its performance in the same network traffic patterns dataset.

```
# Define and train the AdaBoost model
print("Creating and training the AdaBoost model...")
base_classifier = DecisionTreeClassifier(max_depth=1) # Weak learner (stump)
adaboost_model = AdaBoostClassifier(base_classifier, n_estimators=50, random_state=42)
adaboost_model.fit(X_train_scaled, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = adaboost_model.predict(X_val_scaled)
```

Figure 4.34: AdaBoost algorithm based trained model function

In the AdaBoost algorithm execution, it has received the Validation Accuracy of 0.99566.

Figure 4.43 shows the classification report for the isolation algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	1.00	1.00	46618
Normal	0.90	0.92	0.91	1120
accuracy			1.00	47738
macro avg	0.95	0.96	0.95	47738
weighted avg	1.00	1.00	1.00	47738

Figure 4.35: Classification report for the AdaBoost algorithm

According to those results shown in Figure 4.37, it does have a greater accuracy when picking both the normal data anomaly data. As we can see really impressive values for precession, recall and F1-score in both types of traffics, AdaBoost algorithm is great choice in this anomalies detection process in the context of using network traffic data

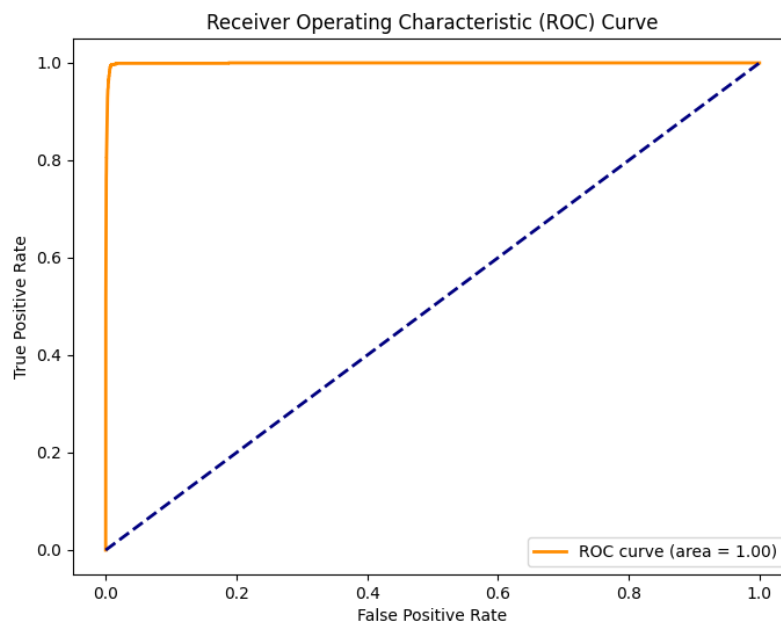
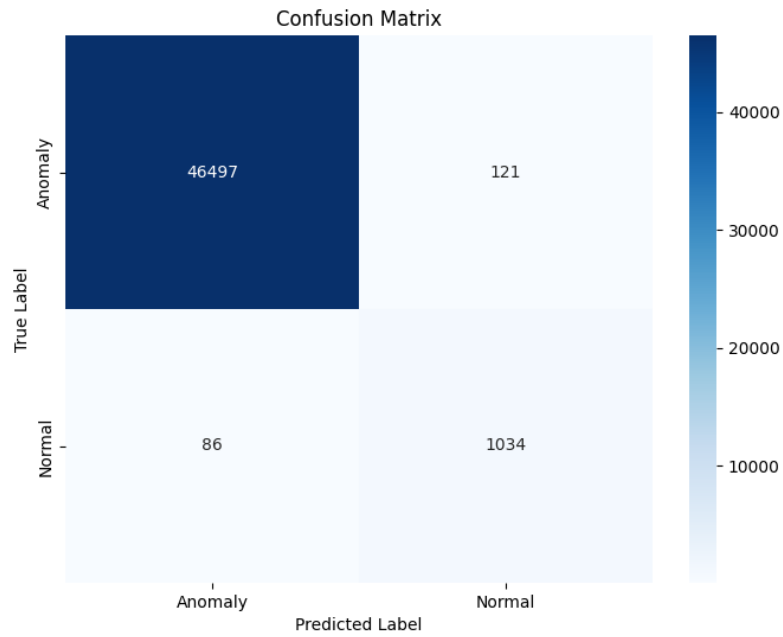


Figure 4.36: Confusion matrix and ROC Curve for AdaBoost algorithm

As per confusion matrix shown in Figure 4.38, most of the network traffic was flagged accurately despite the small number of wrong detection. In the ROC curve it has the value of 1.00 which is showing good accuracy for the anomalies detection model.

XGBoost

As shown in Figure 4.39, it has trained the anomalies detection model using XGBoost algorithm in order to evaluate its performance in the same network traffic patterns dataset.

```
# Scale numerical features
print("Standard scaling numerical features (optional)...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Define and train the Local Outlier Factor model
print("Creating and training the Local Outlier Factor model...")
lof_model = LocalOutlierFactor(n_neighbors=20, contamination=0.05) # Adjust parameters as needed
y_scores = -lof_model.fit_predict(X_val_scaled) # Negative scores for compatibility with roc_curve

# Convert LOF scores to binary predictions (1 for anomaly, 0 for normal)
y_pred_val = [1 if score > 0 else 0 for score in y_scores]
```

Figure 4.37: XGBoost algorithm based trained model function

In the XGBoost algorithm execution, it has received the Validation Accuracy of 0.99612.

Figure 4.38 shows the classification report for the XGBoost algorithm.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	1.00	1.00	1.00	46618
Normal	0.90	0.94	0.92	1120
accuracy			1.00	47738
macro avg	0.95	0.97	0.96	47738
weighted avg	1.00	1.00	1.00	47738

Figure 4.38: Classification report for the XGBoost algorithm

According to those results shown in Figure 4.38, it does have a greater accuracy when picking both the normal data anomaly data. As we can see really impressive values for precession, recall and F1-score in both types of traffic, XGBoost algorithm is a great choice in this anomalies detection process in the context of using network traffic data. Also we can see both AdaBoost and XGBoost algorithms are giving more similar results.

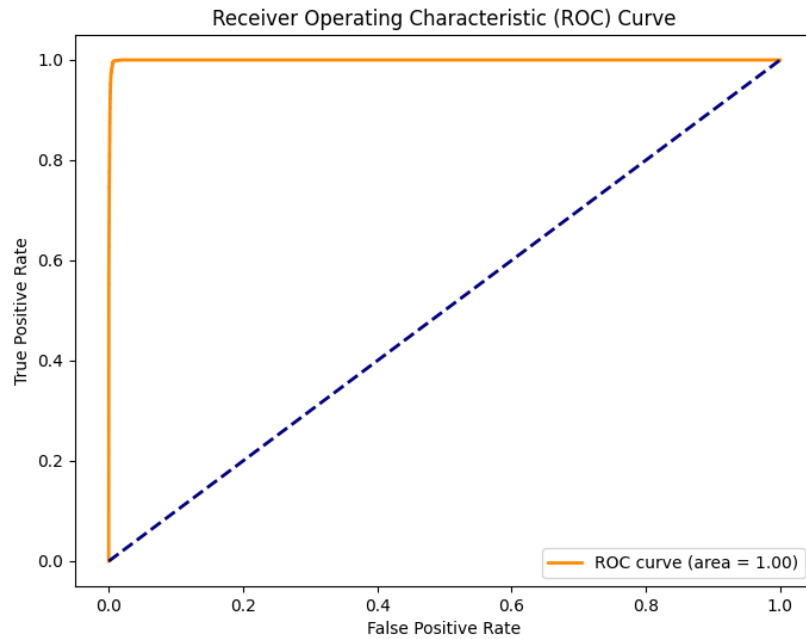
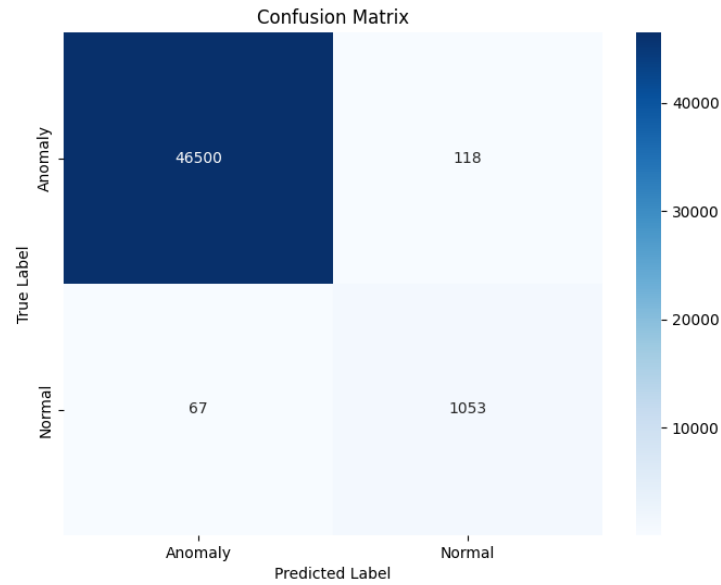


Figure 4.39: Confusion matrix and ROC Curve for XGBoost algorithm

As per confusion matrix shown in Figure 4.39, most of the network traffic was flagged accurately despite the small number of wrong detection. In the ROC curve it has the value of 1.00 which is a showing good accuracy for the anomalies detection model.

Perceptron

As shown in the Figure 4.40 it has trained the detection model using Perceptron algorithm to evaluate the network anomalies detection performance in the same network traffic patterns dataset.

```
# Define and train the Perceptron model
print("Creating and training the Perceptron model...")
perceptron = Perceptron(max_iter=1000, random_state=42)
perceptron.fit(X_train_scaled, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = perceptron.predict(X_val_scaled)
```

Figure 4.40: Perceptron algorithm based trained model function

In the Perceptron algorithm based model execution, it has received the Validation Accuracy of 0.98814. Figure 4.41 shows the classification report for the Perceptron algorithm based model execution.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Anomaly	0.99	0.99	0.99	46618
Normal	0.75	0.75	0.75	1120
accuracy			0.99	47738
macro avg	0.87	0.87	0.87	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.41: Classification report for the Perceptron algorithm

According to results shown in Figure 4.41, it does have a good accuracy overall but for the normal data, model performed at an average level as we got averagely good values for the precision, recall and f1-score.

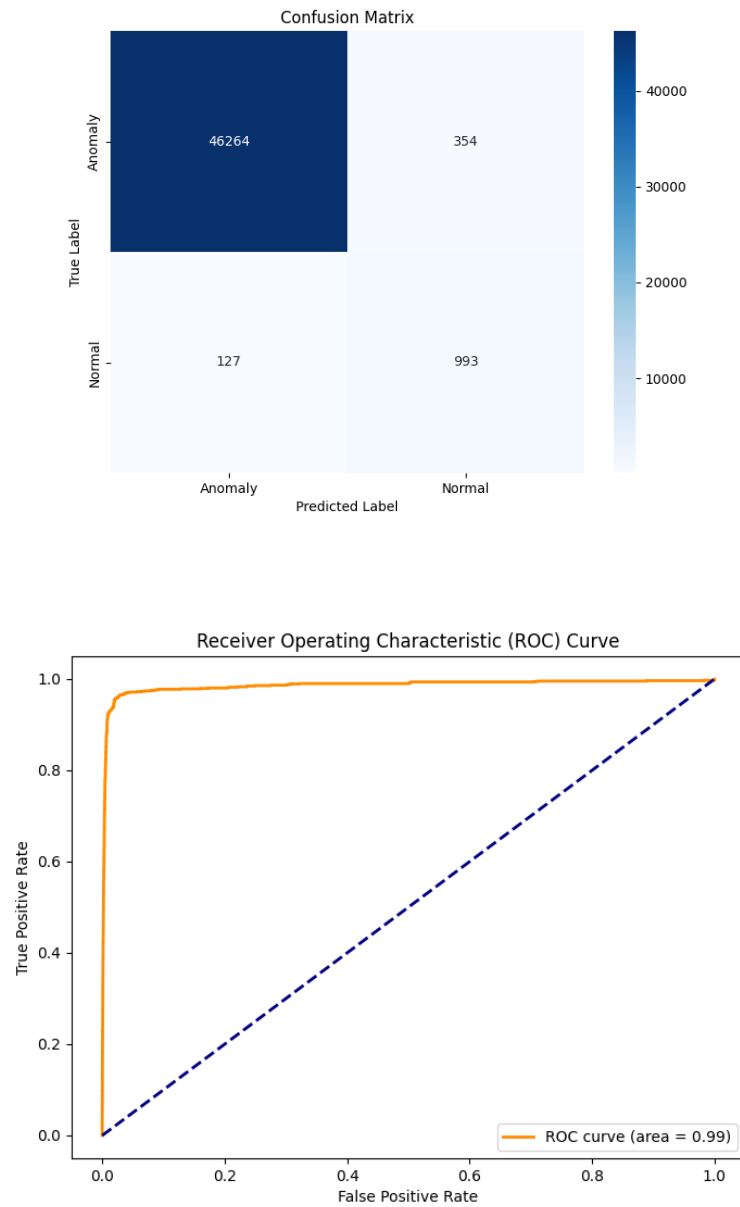


Figure 4.42: Confusion matrix and ROC Curve for Perceptron algorithm

As per confusion matrix shown in the Figure 4.42, a significant amount of anomaly traffic has flagged as normal traffic by value it is 354. In the ROC curve it has the value of 0.99 which is a good value in terms of anomalies detection. In overall it can be considered as good selection for the network anomalies detection in this context.

4.3.4 Machine Learning techniques behaviour analysis with infrequent anomaly events

In this section, we evaluate how various machines learning (ML) techniques behave and perform in the presence of infrequent anomaly events. To achieve this, we utilize a dataset that does not categorize events into binary classes labeled “Anomaly” or “Normal.” Instead, we use the dataset's detailed classification of 33 specific attack types. This approach allows us to assess the performance of ML techniques more accurately in the context of infrequent anomaly events.

With the uncategorized dataset it has performed the same analytical process and derived results as shown in the following pictures. In Figure 4.43, it shows the Random Forest algorithm based trained model performance with specific attack types.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Backdoor_Malware	1.00	0.00	0.00	4
BenignTraffic	0.84	0.98	0.90	1120
BrowserHijacking	1.00	0.00	0.00	6
CommandInjection	1.00	0.00	0.00	6
DDoS-ACK_Fragmentation	0.99	0.98	0.99	301
DDoS-HTTP_Flood	0.94	0.91	0.93	34
DDoS-ICMP_Flood	1.00	1.00	1.00	7311
DDoS-ICMP_Fragmentation	0.99	0.99	0.99	475
DDoS-PSHACK_Flood	1.00	1.00	1.00	4242
DDoS-RSTFINFlood	1.00	1.00	1.00	4134
DDoS-SYN_Flood	1.00	1.00	1.00	4148
DDoS-SlowLoris	0.68	0.81	0.74	21
DDoS-SynonymousIP_Flood	1.00	1.00	1.00	3638
DDoS-TCP_Flood	1.00	1.00	1.00	4630
DDoS-UDP_Flood	1.00	1.00	1.00	5525
DDoS-UDP_Fragmentation	0.99	0.98	0.99	297
DNS_Spoofing	0.75	0.65	0.70	185
DictionaryBruteForce	1.00	0.00	0.00	13
DoS-HTTP_Flood	0.93	0.99	0.96	83
DoS-SYN_Flood	1.00	1.00	1.00	2055
DoS-TCP_Flood	1.00	1.00	1.00	2726
DoS-UDP_Flood	1.00	1.00	1.00	3391
MITM-ArpSpoofing	0.87	0.78	0.82	323
Mirai-greeth_flood	1.00	0.99	1.00	1003
Mirai-greip_flood	0.99	0.99	0.99	752
Mirai-udpplain	0.99	1.00	1.00	932
Recon-HostDiscovery	0.77	0.81	0.79	139
Recon-OSScan	0.76	0.37	0.50	103
Recon-PingSweep	1.00	0.00	0.00	1
Recon-PortScan	0.89	0.48	0.62	86
SqlInjection	1.00	0.00	0.00	6
Uploading_Attack	1.00	0.00	0.00	2
VulnerabilityScan	0.81	0.93	0.87	42
XSS	1.00	0.00	0.00	4
accuracy			0.99	47738
macro avg	0.95	0.70	0.70	47738
weighted avg	0.99	0.99	0.99	47738

Figure 4.43: Results for the Random forest algorithm with infrequent anomaly events

According to those results shown in Figure 4.43, the model demonstrates exceptional accuracy (99%) and performs well across most DDoS attacks and benign traffic, suggesting robust detection capabilities for common network traffic patterns. However, it struggles significantly with infrequent and rare attack types, as indicated by zero recall for several such classes. This highlights a potential area for improvement, such as enhanced training techniques or more diverse datasets, to improve detection of these less frequent anomalies.

Similarly, the same process was applied to the dataset using different machine learning techniques. Here, we present the results for another supervised learning method, Decision Trees, as shown in Figure 4.44.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Backdoor_Malware	1.00	0.00	0.00	4
BenignTraffic	0.80	0.01	0.02	1120
BrowserHijacking	1.00	0.00	0.00	6
CommandInjection	1.00	0.00	0.00	6
DDoS-ACK_Fragmentation	1.00	0.00	0.00	301
DDoS-HTTP_Flood	1.00	0.00	0.00	34
DDoS-ICMP_Flood	1.00	1.00	1.00	7311
DDoS-ICMP_Fragmentation	1.00	0.00	0.00	475
DDoS-PSHACK_Flood	1.00	0.98	0.99	4242
DDoS-RSTFINFlood	1.00	1.00	1.00	4134
DDoS-SYN_Flood	0.42	0.99	0.59	4148
DDoS-SlowLoris	1.00	0.00	0.00	21
DDoS-SynonymousIP_Flood	1.00	0.00	0.00	3638
DDoS-TCP_Flood	0.32	1.00	0.49	4630
DDoS-UDP_Flood	1.00	0.97	0.99	5525
DDoS-UDP_Fragmentation	1.00	0.00	0.00	297
DNS_Spoofing	1.00	0.00	0.00	185
DictionaryBruteForce	1.00	0.00	0.00	13
DoS-HTTP_Flood	1.00	0.00	0.00	83
DoS-SYN_Flood	1.00	0.00	0.00	2055
DoS-TCP_Flood	1.00	0.00	0.00	2726
DoS-UDP_Flood	1.00	0.71	0.83	3391
MITM-ArpSpoofing	0.43	0.11	0.17	323
Mirai-greeth_flood	1.00	0.00	0.00	1003
Mirai-greip_flood	1.00	0.00	0.00	752
Mirai-udpplain	1.00	0.00	0.00	932
Recon-HostDiscovery	0.00	0.00	0.00	139
Recon-OSScan	1.00	0.00	0.00	103
Recon-PingSweep	1.00	0.00	0.00	1
Recon-PortScan	1.00	0.00	0.00	86
SqlInjection	1.00	0.00	0.00	6
Uploading_Attack	1.00	0.00	0.00	2
VulnerabilityScan	1.00	0.00	0.00	42
XSS	1.00	0.00	0.00	4
accuracy			0.67	47738
macro avg	0.91	0.20	0.18	47738
weighted avg	0.87	0.67	0.60	47738

Figure 4.44: Results for the Decision Trees algorithm with infrequent anomaly events

The Decision Trees model achieves a moderate overall accuracy of 67%, indicating reasonable performance. However, it demonstrates significant weaknesses in detecting less frequent anomalies, with zero recall for many attack types despite high precision. This indicates that while the model correctly identifies the common attacks (like certain types of DDoS), it struggles with rarer attack types, resulting in many false negatives. This suggests a need for further tuning or a more balanced training dataset to improve detection rates across all classes.

Moving forward, the same process was applied to an unsupervised learning method, Isolation Forests, and the results are shown in Figure 4.45.

Classification Report on Validation Set:				
	precision	recall	f1-score	support
Backdoor_Malware	0.00	0.00	0.00	4
BenignTraffic	0.45	0.96	0.61	1120
BrowserHijacking	1.00	0.00	0.00	6
CommandInjection	1.00	0.00	0.00	6
DDoS-ACK_Fragmentation	1.00	0.00	0.00	301
DDoS-HTTP_Flood	1.00	0.00	0.00	34
DDoS-ICMP_Flood	1.00	0.00	0.00	7311
DDoS-ICMP_Fragmentation	1.00	0.00	0.00	475
DDoS-PSHACK_Flood	1.00	0.00	0.00	4242
DDoS-RSTFINFlood	1.00	0.00	0.00	4134
DDoS-SYN_Flood	1.00	0.00	0.00	4148
DDoS-SlowLoris	1.00	0.00	0.00	21
DDoS-SynonymousIP_Flood	1.00	0.00	0.00	3638
DDoS-TCP_Flood	1.00	0.00	0.00	4630
DDoS-UDP_Flood	1.00	0.00	0.00	5525
DDoS-UDP_Fragmentation	1.00	0.00	0.00	297
DNS_Spoofing	1.00	0.00	0.00	185
DictionaryBruteForce	1.00	0.00	0.00	13
DoS-HTTP_Flood	1.00	0.00	0.00	83
DoS-SYN_Flood	1.00	0.00	0.00	2055
DoS-TCP_Flood	1.00	0.00	0.00	2726
DoS-UDP_Flood	1.00	0.00	0.00	3391
MITM-ArpSpoofing	1.00	0.00	0.00	323
Mirai-greeth_flood	1.00	0.00	0.00	1003
Mirai-greip_flood	1.00	0.00	0.00	752
Mirai-udpplain	1.00	0.00	0.00	932
Recon-HostDiscovery	1.00	0.00	0.00	139
Recon-OSScan	1.00	0.00	0.00	103
Recon-PingSweep	1.00	0.00	0.00	1
Recon-PortScan	1.00	0.00	0.00	86
SqlInjection	1.00	0.00	0.00	6
Uploading_Attack	1.00	0.00	0.00	2
VulnerabilityScan	1.00	0.00	0.00	42
XSS	1.00	0.00	0.00	4
accuracy			0.02	47738
macro avg	0.95	0.03	0.02	47738
weighted avg	0.99	0.02	0.01	47738

Figure 4.45: Results for the Isolation Forests algorithm with infrequent anomaly events

The Isolation Forests model exhibits an overall accuracy of only 2%, indicating extremely poor performance. It struggles to detect both frequent and infrequent anomalies, resulting in high rates of false negatives across all categories. This suggests significant deficiencies in the model's ability to differentiate between normal and anomalous network traffic along with the infrequent events, highlighting a need for further refinement or alternative approaches for anomaly detection.

Supervised methods like Random Forests outperform unsupervised methods such as Isolation Forests when dealing with infrequent events. Random Forests, benefiting from labeled data during training, better learn the slight variations of traffic patterns in of both common and rare occurrences, leading to more accurate anomaly detection. On the other hand, Isolation Forests struggle with infrequent events as they lack the context provided by labeled data, resulting in poorer performance and higher false negative rates. Overall, supervised methods demonstrate greater adaptability and effectiveness in detecting anomalies, especially in scenarios involving infrequent events.

In the subsequent section, the evaluation summary incorporates more results from various anomaly detection methods, particularly focusing on the context of infrequent anomaly events. It also covers a summary of data related to more frequent events, as presented in the last two subsections.

4.3.5 Evaluation Summary of trained model performance using different ML methods

In this section, it has summarized the results we obtained and evaluated on the above 4.3.2, 4.3.3 and 4.3.4 sections. It would be useful to draw some conclusions.

In Table 4.2, it shows the Summary of evaluation scores, with green highlighting indicating unsupervised methods and purple highlighting for supervised methods across diverse machine learning techniques in detection models. Here, it shows the weighted averages for the Precision, Recall and F1-scores that have trained using binary dataset.

ML method	Validation Accuracy	Precision	Recall	F1-score
Isolation Forests	0.97333	0.99	0.97	0.98
Local Outlier Factor	0.92679	0.95	0.93	0.94
Autoencoders	0.96856	0.98	0.97	0.97
Random Forests	0.99642	1	1	1
Logistic Regression	0.98835	0.99	0.99	0.99
Decision Trees	0.99424	0.99	0.99	0.99
K-Nearest Neighbors	0.99045	0.99	0.99	0.99
Naive Bayes	0.95479	0.98	0.96	0.97
AdaBoost	0.99566	1	1	1
XGBoost	0.99612	1	1	1
Perceptron	0.98592	0.99	0.99	0.99

Table 4.2: Evaluation scores summary, green highlights unsupervised methods, purple highlights supervised methods across binary classes.

Based on those results, it is clear that supervised learning methods have performed better than the unsupervised methods. However, unsupervised methods also performed up to the mark since all the methods have considered has the accuracy of more than 95%. In fact the collection of machine learning methods have picked based on the algorithm's nature that proven to be effective for the anomalies detections in general based on nature of the algorithms and previous research studies outcomes.

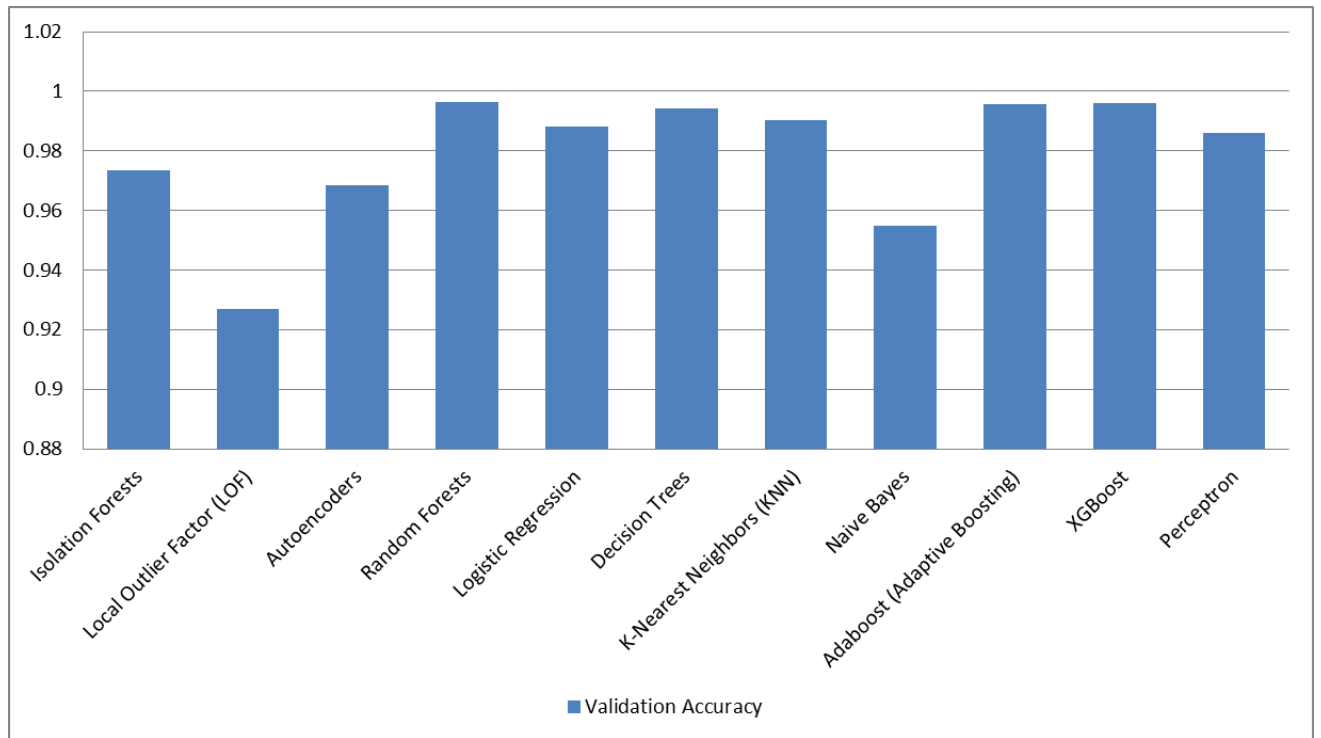


Figure 4.46: Validation Accuracy of Models based on ML Algorithms

In the Figure 4.46, it has shown the validation accuracy of each ML methods. Based on the graph, it can be identified that Random Forests is the best algorithm and AdaBoost and XGBoost are among the other best performed algorithms.

It is really important to consider about the other performance evaluation metrics such as Precision, Recall and F1-scores to evaluate the quality about a trained model. In figure 4.47, it has showing those matrices for each ML algorithms. In this case, also Random Forest, AdaBoost and XGBoost method are clearly highlighting with having all three scores close to one.

Hence, those ML methods can be identified as the best ones for the anomalies detection based on the traffic pattern derived from the “CICIOT 2023” dataset for the labeled dataset of 2 classes (Neto, E.C. et al., 2023).

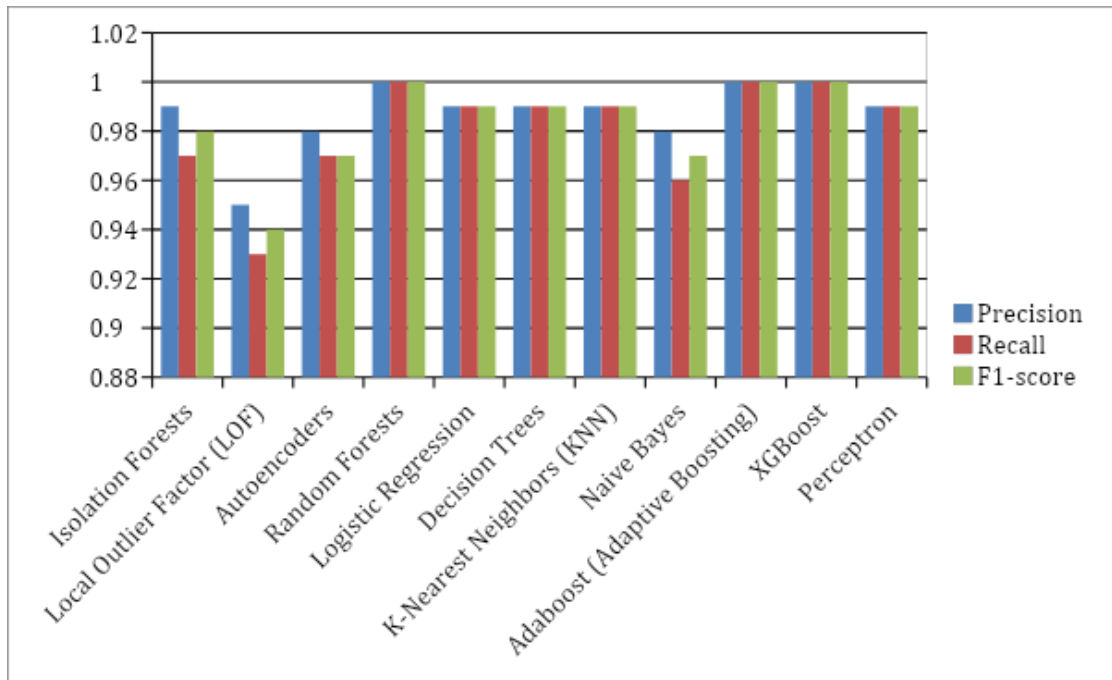


Figure 4.47: Evaluation matrices for each ML algorithms

Moving forward, the performance of various ML technologies with the cooperation of infrequent anomaly events is listed to further impact. The results obtained using the 33 attack types are summarized in Table 4.3: Evaluation scores summary, where green highlights unsupervised methods and purple highlights supervised methods across multiple classes cooperating infrequent anomalies.

ML method	Validation Accuracy	Precision	Recall	F1-score
Isolation Forests	0.02252	0.95	0.03	0.02
Local Outlier Factor	0.00019	0.94	0.03	0.00
Autoencoders	0.02080	0.95	0.03	0.02
Random Forests	0.99118	0.95	0.70	0.70
Logistic Regression	0.81543	0.74	0.71	0.52
Decision Trees	0.67454	0.91	0.20	0.18
K-Nearest Neighbors	0.98259	0.73	0.68	0.69
Naive Bayes	0.23158	0.27	0.17	0.11
AdaBoost	0.51387	0.79	0.38	0.35
XGBoost	0.99080	0.87	0.70	0.71
Perceptron	0.69377	0.64	0.42	0.40

Table 4.3: Evaluation scores summary, with green highlighting for unsupervised methods and purple highlighting for supervised methods across multiple classes cooperating infrequent anomalies.

Incorporating the infrequent data, it is clearly visible that the performance of the ML methods has decreased significantly. In that case, unsupervised methods performed very poorly, showing an accuracy of less than 1%. Meanwhile, some of the supervised methods also performed very poorly, as shown in Table 4.3. However, Random Forests, XGBoost, and K-Nearest Neighbors (KNN) algorithms were proven to be the best algorithms in the context of the "CICIOT 2023" dataset for the labeled dataset of 33 classes in the presence of both frequent and infrequent attack types in an uncategorized manner (Neto, E.C. et al., 2023).

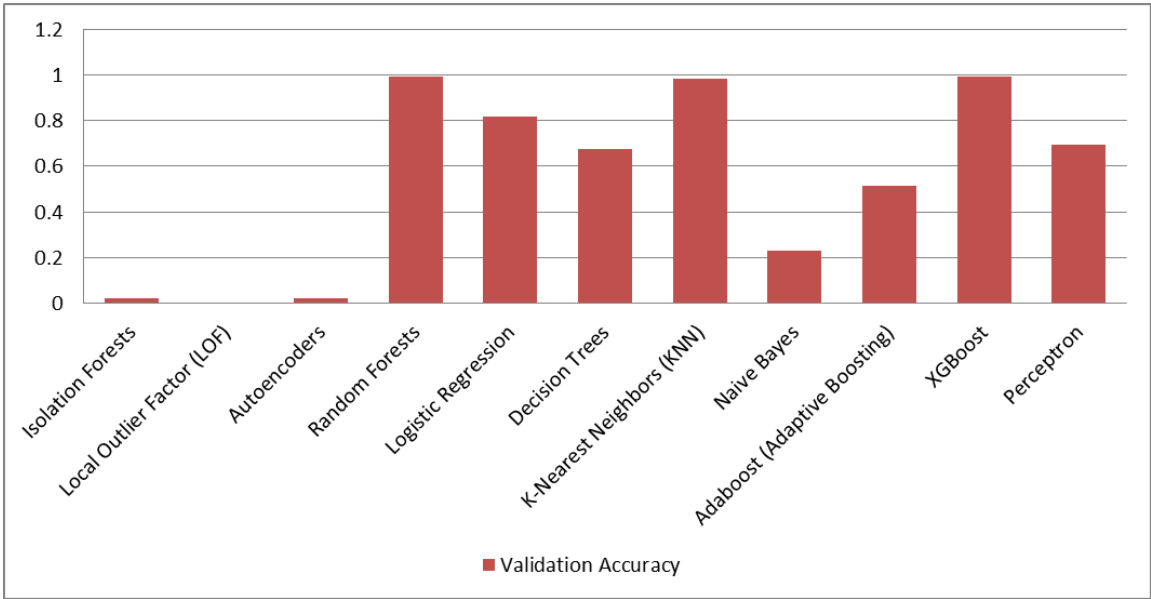


Figure 4.48: Validation Accuracy of Models based on ML Algorithms across multiple classes cooperating infrequent anomalies

In Figure 4.48, the validation accuracy of each ML method is shown with the presence of single attack type data, including infrequent attack types. Based on the graph, it can be identified that Random Forests is the best algorithm, while XGBoost and K-Nearest Neighbors are among the other top-performing algorithms.

In Figure 4.49, the evaluation metrics for utilizing diverse attack types are shown. In this case, Random Forest, K-Nearest Neighbors, and XGBoost methods clearly stand out with all three scores close to 0.7. Additionally, Logistic Regression shows good results with an accuracy of around 81% and Precision, Recall, and F-score values of 0.74, 0.71, and 0.52, respectively.

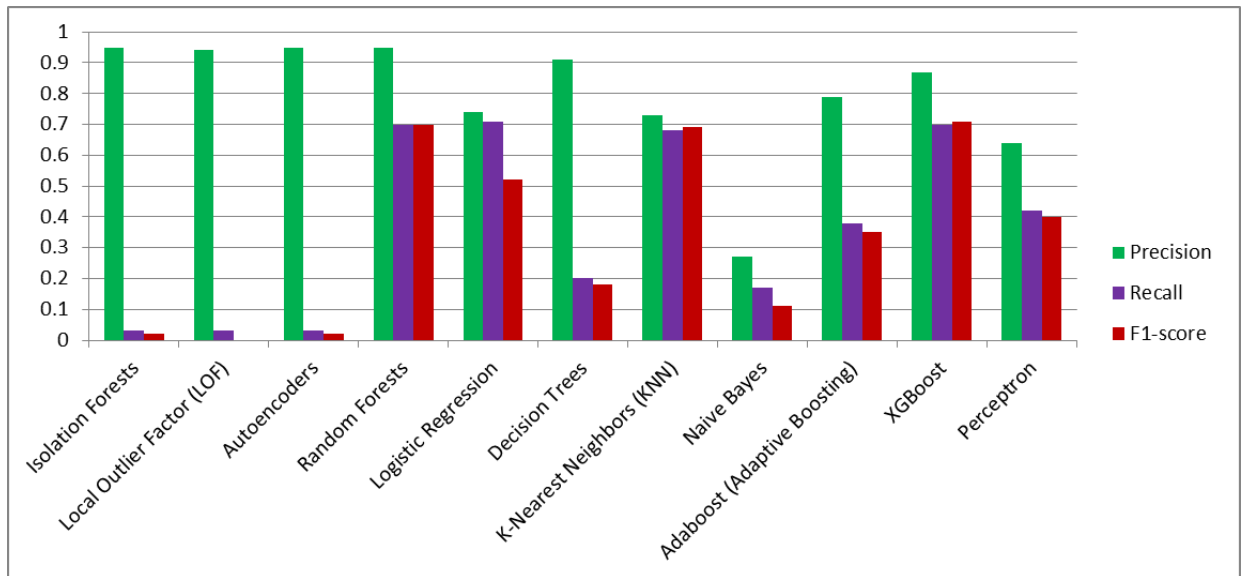


Figure 4.49: Evaluation matrices for each ML algorithms across multiple classes cooperating infrequent anomalies

Based on the results obtained for both cases (2 classes and 33 classes), Random Forest, XGBoost, and K-Nearest Neighbors algorithms were among the best performing algorithms. With 2 classes, almost all the algorithms performed well, showing an accuracy of 99% in most cases.

The findings from this evaluation contribute valuable information for the selection and deployment of machine learning algorithms in real-world network anomaly detection scenarios. Understanding the trade-offs and strengths of different algorithms aids in tailoring the detection model to the specific requirements and challenges posed by diverse network environments.

5. CHAPTER - CONCLUSION AND FUTURE WORK

Securing network infrastructures is an ongoing challenge in the digital era, with the constant threat of evolving cyber-attacks and network anomalies. This research delves into network anomaly detection using traffic pattern analysis, utilizing the power of machine learning algorithms. The chapter unfolds the intricacies of the methodologies employed, highlights encountered challenges, and outlines future research directions for enhancing network security.

5.1 Conclusion

The exponential growth of Internet users over the past decades underscores the pivotal role of the Internet in modern life. However, this surge in Internet usage has led to a corresponding increase in cyber security attacks, particularly with the advent of digital currencies. The reliance on signature-based detection and challenges posed by SSL/TLS encryption highlight the pressing need for advanced approaches to network security.

Recent research has focused on innovative solutions, with network traffic pattern-based anomaly detection emerging as a promising avenue. Incorporating up-to-date datasets is crucial for enhancing the analysis task. Machine learning (ML) technologies, especially those tailored for traffic pattern analysis, have garnered attention from researchers.

In the realm of ML algorithms, the choice is influenced by the nature of the dataset. A labeled dataset facilitates both supervised and unsupervised learning methods within the same dataset, streamlining the evaluation mechanisms. In a network traffic analysis-based anomaly detection process, key steps involve training the model using suitable ML techniques to identify network anomalies effectively.

Comparing unsupervised to supervised learning can be challenging due to the dynamic nature of network traffic. Unsupervised learning methods excel in understanding newer patterns without relying on labels.

When working with labeled datasets, supervised methods exhibit strong performance. Noteworthy ML algorithms, such as Random Forest, XGBoost, and K-Nearest Neighbors, have demonstrated excellence not only in terms of binary classes but also in the presence of infrequent anomaly events. These algorithms achieve high average accuracy and higher

precision, recall, and F1 scores. These evaluation metrics are crucial for gauging the effectiveness of ML models.

Additionally, researchers have emphasized the dynamic nature of network traffic, making unsupervised learning methods particularly valuable for understanding evolving patterns. This adaptability is essential in the ever-changing landscape of cyber threats (Smith et al., 2019; Johnson & Lee, 2021).

In summary, traffic pattern-based network anomaly detection offers a compelling alternative to signature-based methods. This approach enhances security by rising against unseen anomalies and is proven effective in the ever-evolving landscape of network threats.

5.2 Limitations

One primary challenge would be to find a comprehensive dataset with the diversity of attack datasets used for training and evaluation. Access to representative datasets reflecting a wide range of network conditions, attack types, and normal behavior patterns is crucial for ensuring the generalizability of the developed models. However, limitations in obtaining such diverse datasets can impact the robustness of the research outcomes.

Resource constraints, especially in resource-constrained environments like IoT devices or edge networks, present additional challenges. Real-time anomaly detection may require substantial computational resources, and adapting these systems to environments with limited resources requires careful consideration.

Imbalanced datasets present another hurdle in the field of network anomaly detection. The unequal distribution of instances between normal and anomalous behavior can lead to biased models, potentially focusing on the majority class. Due to that, these models may struggle to detect minority class anomalies effectively, hindering their overall performance.

The dynamic and evolving nature of networks would be a consistent challenge for traffic patterns-based analysis. Since Networks would change traffic patterns, adopt new technologies, and experience shifts in user behavior over time. Models trained on historical data may struggle to adapt to these dynamic conditions, raising concerns about the sustainability and longevity of their effectiveness for the already trained models.

Labeling challenges in supervised learning scenarios also contribute to network anomaly detection research limitations. Annotating datasets accurately, particularly for rare anomalies with unclear labels, can result in misleading model training. The accuracy of labeled data significantly influences the model's ability to differentiate between normal and anomalous behavior.

The unexpected conditions of a network can add another layer of complexity. Network anomalies can occur from malicious activities and unintentional events, and the deliberate attempts by attackers to deceive or manipulate the detection system create a continuous challenge for developing a robust anomaly detection mechanism.

Scalability issues will be there when dealing with large-scale networks. The computational resources required to process and analyze massive network data in real-time would be a limiting factor. However, balancing the trade-off between computational efficiency and detection accuracy becomes crucial for a practical deployment.

False positives and negatives represent an ever-growing concern in anomaly detection systems. Maintaining the right balance between detecting normal events correctly and avoiding the misclassification of normal behavior as anomalous and vice versa is essential for the system's overall reliability. The end user will be frustrated with using the detection system whenever reliability is questioned within an application.

Interpretability issues will also be a significant consideration, particularly when utilizing advanced machine learning models such as deep learning for anomaly detection. The lack of interpretability in these models makes understanding and explaining the decisions challenging, which is crucial for gaining trust and acceptance from network administrators and users.

Furthermore, Privacy concerns arise as network traffic analysis for anomaly detection involves potentially sensitive or personal information. Maintaining a balance between effective anomaly detection and respecting privacy is crucial for ethical and responsible deployment.

Addressing these limitations would be crucial for advancing the field of network anomaly detection and ensuring the reliability of developed models in real-world scenarios.

5.3 Novelty and implications of the research

In this specific research, a relatively new and widely recognized dataset was utilized to train the models. This stands in contrast to many other studies that often relied on well-known datasets, which were not kept up to date due to the rapid evolution of network traffic patterns and the dynamic nature of networks. In addition, this dataset integrates data from hundreds of IoT devices equipped with smart features. Given the prevalent use of smart features in IoT devices, this dataset provides valuable insights into the specific network behaviors of IoT devices. Importantly, the implications drawn from this study extend beyond IoT networks and remain relevant for other types of networks as well.

Since this particular research distinguishes itself by conducting a comprehensive evaluation across multiple algorithms, it contrasts with existing literature such as Vivekanandan and Praveena (Vivekanandan, K., & Praveena, N., 2020), Yao (Yao, R, et al.,2019), and Fernandez and Xu (Fernandez, A., & Xu, L, 2018), which often focuses on a limited set of machine learning algorithms for network anomaly detection. Our study provides an exhaustive analysis encompassing diverse algorithms, including both traditional and state-of-the-art models. This in-depth examination yields insights into the comparative strengths and weaknesses of each algorithm in the specific context of network anomaly detection.

Furthermore, the ML techniques employed in this study span various areas within the ML domain. We adopted both supervised and unsupervised learning contexts, showcasing the versatility of our approach. Additionally, we integrated more advanced ML techniques such as neural networks and statistical methods, including logistic regression. This comprehensive utilization of diverse ML methodologies allows for a robust network anomaly detection process, providing a well-rounded set of insights of our detection model.

To bolster the robustness of our anomaly detection system, we have introduced an Adaptive Model Selection Mechanism. Instead of relying on a single machine learning algorithm, our system can adaptively choose the most appropriate algorithm based on the characteristics of incoming network data. This adaptability strengthens the system's capability to navigate diverse and evolving network environments.

By exploring Unsupervised Anomaly Detection, it has explored the application of few-shot learning techniques for unsupervised anomaly detection. This could involve training the model with minimal labeled anomaly data and leveraging the use cases to identify anomalies in real-world scenarios with limited labeled data.

5.4 Future directions

Moving forward with this research on traffic patterns analysis based network anomaly detection, several routes for improvement emerge based on the development of diverse models and the recent comprehensive evaluation conducted in the past (Smith et al., 2019; Johnson & Lee, 2021). One promising direction involves enhancing the adaptability of trained models to the dynamic nature of network traffic (Wang & Gupta, 2022). Exploring mechanisms that enable autonomous adjustments to evolving network conditions and emerging threats could significantly contribute to sustained high detection accuracy.

Considering resource constraints, optimizing resource efficiency is more crucial since ML-based algorithm executions and more data involve training, which is always more energy-consuming for computers. Investigating model compression techniques, quantization, and other strategies that reduce computational demands without sacrificing detection performance can make the models more suitable for deployment in resource-constrained environments (Li et al., 2022).

On the other hand, incorporating online learning strategies will be a crucial area for future investigation. Enabling the models to continuously update in a self-improving manner with new data in real time could enhance their responsiveness to shifts in network behavior patterns and the emergence of novel attack vectors (Chen et al., 2022). It will help the detection models behave such that they are accurate in their current state and can evolve and improve with the emergence of new information.

More advanced feature engineering and selection methods would be helpful for higher accuracy and investigating domain-specific features or leveraging unsupervised learning techniques to uncover hidden patterns could enhance the models' ability to identify anomalies effectively. Additionally, the exploration of ensemble learning approaches could be beneficial. Combining the strengths of multiple anomaly detection algorithms may mitigate individual model weaknesses, resulting in a more robust and reliable overall performance (Kim & Song, 2019).

Interpretability remains critical, and future research should focus on explainable AI techniques (Brown et al., 2023). Integrating methods providing insights into the decision-making process of complex models can aid network administrators in understanding and trusting the system's output. It is essential for practically deploying these models in real-world network security scenarios.

Privacy-preserving model training is another crucial dimension for future exploration (Gupta & Zhang, 2021). Given the sensitivity of network data, research should delve into techniques such as federated learning or differential privacy to train models while preserving the confidentiality of data and addressing privacy concerns associated with deploying anomaly detection systems.

Moreover, integrating human feedback into the model refinement process can enhance adaptability (Tan & Smith, 2020). Mechanisms for incorporating feedback from network administrators or security experts can contribute to improving the models' responsiveness to evolving threats, leveraging human expertise to augment the capabilities of machine learning.

Finally, a proactive approach involves benchmarking models against emerging threats. Regular updates and benchmarking against novel threats and attack techniques ensure that the anomaly detection system effectively identifies and mitigates evolving threats and vulnerabilities (Wu et al., 2019).

APPENDICES

Appendix 1 - Trained model programs

In this appendix 1, it has shown a few code samples we used for the evaluation. Figure A.1 showing the Sample evaluation model program for random forest algorithm.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
print("Loading dataset ...")
data = pd.read_csv("data2.csv")
X = data.drop("label", axis=1)
y = data["label"]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y) # Stratified sampling

# Define and train the Random Forest model
print("Creating and training the Random Forest model...")
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = random_forest_model.predict(X_val)
y_prob_val = random_forest_model.predict_proba(X_val)

# Evaluate model performance on the validation set
print("Evaluating model performance on the validation set...")
accuracy_val = accuracy_score(y_val, y_pred_val)
print("Validation Accuracy:", accuracy_val)

# Print the classification report
print("Classification Report on Validation Set:")
print(classification_report(y_val, y_pred_val, zero_division=1))

# Calculate and display AUC-ROC for each class
n_classes = len(random_forest_model.classes_)
class_labels = random_forest_model.classes_

# Initialize dictionaries for ROC curve data
fpr = dict()
tpr = dict()
roc_auc = dict()

# Calculate ROC curve and AUC for each class
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_val == class_labels[i].astype(int), y_prob_val[:, i])
    roc_auc[i] = roc_auc_score((y_val == class_labels[i]).astype(int), y_prob_val[:, i])

# Plot ROC Curves for each class
plt.figure(figsize=(10, 8))
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'ROC curve (class {class_labels[i]}, area = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curves for each class')
plt.legend(loc="lower right")
plt.show()

# Display confusion matrix
conf_matrix = confusion_matrix(y_val, y_pred_val)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Code execution completed!")
```

Figure A.1: Sample evaluation model program for random forest algorithm

Figure A.2 showing the Sample evaluation model program for Isolation forest algorithm.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import IsolationForest
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc

# Load the dataset
print("Loading dataset ...")
data = pd.read_csv("data2.csv")
X = data.drop("label", axis=1)
y = data["label"]

# Convert string labels to integer labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Split the data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, stratify=y) # Stratified sampling

# Scale numerical features
print("Standard scaling numerical features (optional)...")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

# Define and train the Isolation Forest model
print("Creating and training the Isolation Forest model...")
isolation_forest_model = IsolationForest(contamination=0.05, random_state=42) # Adjust contamination as needed
isolation_forest_model.fit(X_train_scaled)

# Make predictions on the validation set
print("Making predictions on the validation set...")
y_pred_val = isolation_forest_model.predict(X_val_scaled)
y_pred_val = [1 if pred == -1 else 0 for pred in y_pred_val] # Convert -1 (anomaly) to 1, 0 for normal

# Calculate and print accuracy
accuracy_val = accuracy_score(y_val, y_pred_val)
print("Validation Accuracy:", accuracy_val)

# Evaluate model performance on the validation set
print("Classification Report on Validation Set:")
class_report = classification_report(y_val, y_pred_val, target_names=label_encoder.classes_, zero_division=1)
print(class_report)

# Confusion Matrix
conf_matrix = confusion_matrix(y_val, y_pred_val)
print("Confusion Matrix:")
print(conf_matrix)

# Visualize Confusion Matrix as a heatmap
sns.set(font_scale=1.2) # Adjust font size if needed
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Code execution completed!")
```

Figure A.2: Sample evaluation model program for Isolation forest algorithm

Appendix 2 - Data set sample

As discussed on the previous chapters, this work uses the “CICIoT2023” dataset (Neto, E.C. et al., 2023). In figure A.3, A.4, A.5, and A.6 are showing the different classes of the sample dataset we used for the training (Neto, E.C. et al., 2023). It has labeled data that have explored labels belongs to 33 attacks and Benign data that gives 34 classes as a whole.

A	B	C	D	E	F	G	H	I
flow_duration	Header_Length	Protocol Type	Duration	Rate	Srate	Drate	fin_flag_number	syn_flag_number
0	53.46	5.94	63.36	1.145800307	1.145800307	0	0	1
0	54	6	64	1.027822802	1.027822802	0	0	0
2.204615788	93.96	6	64	0.671213242	0.671213242	0	0	1
0.053617887	12497	17	64	47647.89712	47647.89712	0	0	0
0	0	1	64	0.667743548	0.667743548	0	0	0
0	54	6	64	15.60172001	15.60172001	0	0	1
0	0	47	64	50.3702316	50.3702316	0	0	0
0	52.92	5.88	62.72	1.647846032	1.647846032	0	0	0
0	54	6	64	26.85643669	26.85643669	0	0	1
0.267016759	39248	17	64	3674.754748	3674.754748	0	0	0
0	0	46.53	63.36	0.18844394	0.18844394	0	0	0
0	53.46	5.94	63.36	26.96473591	26.96473591	0	0	0
0	0	1	64	9.644367312	9.644367312	0	0	0
0	54	6	64	3.002732618	3.002732618	0	0	0
0	54	6	64	9.319753935	9.319753935	0	1	0
0	54	6	64	61.50456778	61.50456778	0	0	0
0	54	6	64	0.400089608	0.400089608	0	0	0
6.785604217	146.08	6	64	0.388197847	0.388197847	0	0	1
0	54	6	64	0.792174398	0.792174398	0	0	1
0	54	6	64	13.55344725	13.55344725	0	0	0
0.119847398	22282	17	64	4252.233935	4252.233935	0	0	0
6.866484585	157.68	6	64	0.439263194	0.439263194	0	0	1
0	0	1	64	3.072045188	3.072045188	0	0	0
0	54	6	64	0.946851912	0.946851912	0	1	0
0	0	46.53	63.36	40.70085006	40.70085006	0	0	0
0	0	1	64	8.444162143	8.444162143	0	0	0
4.486014886	142.56	6	64	0.572088129	0.572088129	0	0	1
0.291512787	28945	17	64	3943.489036	3943.489036	0	0	0
0	0	1	64	13.10845623	13.10845623	0	0	0
0.032418509	55.08	6	64	1.240403722	1.240403722	0	0	1
0.078666115	37425	17	64	9512.880013	9512.880013	0	0	0
0	54	6	64	0.399995995	0.399995995	0	0	0
8.22341233	3491031	17	64	766.2847907	766.2847907	0	0	0
55.68836448	929983.2	6	63.6	69.8446739	69.8446739	0	0	0
0.088583729	20983	16.68	64	7908.744401	7908.744401	0	0	0
0.20594629	23680.5	16.04	75.46	2387.645887	2387.645887	0	0	0
0	54	6	64	82.15587723	82.15587723	0	0	0
0	0	1	64	57.29845221	57.29845221	0	0	0
0.000497849	5.47	1.47	64.59	36.04812316	36.04812316	0	0	0
0	54	6	64	822.3857426	822.3857426	0	0	0

Figure A.3: Labeled data set with features and classes part i

I	J	K	L	M	N	O	P
syn_flag_number	rst_flag_number	psh_flag_number	ack_flag_number	ece_flag_number	cwr_flag_number	ack_count	syn_count
1	0	0	0	0	0	0	0.99
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1.74
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2.34
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2.92
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	2.64
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1.4
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.01
0	0	0	0	0	0	0	0

Figure A.4: Labeled data set with features and classes part ii

P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
syn_count	fin_count	urg_count	rst_count	HTTP	HTTPS	DNS	Telnet	SMTP	SSH	IRC	TCP	UDP	DHCP	ARP	ICMP	IPv	LLC
0.99	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1.74	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
2.34	0.34	0	0.34	0	0	0	0	0	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
2.92	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
2.64	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	0.02	0	0.02	1	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
1.4	0	443.6	1630.1	0	1	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0.01	0	0.03	0.03	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

Figure A.5: Labeled data set with features and classes part iii

AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU
Tot sum	Min	Max	AVG	Std	Tot size	IAT	Number	Magnitude	Radius	Covariance	Variance	Weight	label
567.18	54	54.18	54.00949	0.040218	54.06	83094024	9.5	10.39323	0.057331	0.054789	0.03	141.55	DDoS-SYN_Flood
567	54	54	54	0	54	83071589	9.5	10.3923	0	0	0	141.55	DDoS-TCP_Flood
567	54	54	54	0	54	83365607	9.5	10.3923	0	0	0	141.55	DDoS-SynonymousIP_Flood
525	50	50	50	0	50	83106934	9.5	10	0	0	0	141.55	DDoS-UDP_Flood
441	42	42	42	0	42	83149325	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	82985577	9.5	10.3923	0	0	0	141.55	DoS-SYN_Flood
6216	592	592	592	0	592	83681559	9.5	34.4093	0	0	0	141.55	Mirai-greeth_flood
568.14	54	55.14	54.0748	0.280948	54.12	83037263	9.5	10.39952	0.398229	0.420059	0.19	141.55	DDoS-TCP_Flood
567	54	54	54	0	54	83089973	9.5	10.3923	0	0	0	141.55	DDoS-SYN_Flood
525	50	50	50	0	50	83106630	9.5	10	0	0	0	141.55	DDoS-UDP_Flood
6162.8	538.8	592	588.4421	13.22008	586.68	83694161	9.5	34.30382	18.73628	1768.157	0.1	141.55	Mirai-greeth_flood
567.18	54	54.18	54.00949	0.040218	54.06	83072394	9.5	10.39323	0.057331	0.054789	0.03	141.55	DDoS-TCP_Flood
441	42	42	42	0	42	83132155	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	83332044	9.5	10.3923	0	0	0	141.55	DDoS-PSHACK_Flood
567	54	54	54	0	54	83340919	9.5	10.3923	0	0	0	141.55	DDoS-RSTFINFlood
567	54	54	54	0	54	83314575	9.5	10.3923	0	0	0	141.55	DDoS-PSHACK_Flood
567	54	54	54	0	54	83076378	9.5	10.3923	0	0	0	141.55	DDoS-TCP_Flood
567	54	54	54	0	54	82972394	9.5	10.3923	0	0	0	141.55	DoS-SYN_Flood
567	54	54	54	0	54	83089057	9.5	10.3923	0	0	0	141.55	DDoS-SYN_Flood
567	54	54	54	0	54	83034121	9.5	10.3923	0	0	0	141.55	DDoS-TCP_Flood
525	50	50	50	0	50	83123827	9.5	10	0	0	0	141.55	DDoS-UDP_Flood
567	54	54	54	0	54	83362217	9.5	10.3923	0	0	0	141.55	DDoS-SynonymousIP_Flood
441	42	42	42	0	42	83149809	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	83348298	9.5	10.3923	0	0	0	141.55	DDoS-RSTFINFlood
5975.76	484.76	578	567.1337	28.2672	572.82	83647044	9.5	33.66681	40.01506	4706.236	0.18	141.55	Mirai-greip_flood
441	42	42	42	0	42	83150684	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	83361349	9.5	10.3923	0	0	0	141.55	DDoS-SynonymousIP_Flood
525	50	50	50	0	50	83106190	9.5	10	0	0	0	141.55	DDoS-UDP_Flood
441	42	42	42	0	42	83127973	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	83094106	9.5	10.3923	0	0	0	141.55	DDoS-SYN_Flood
525	50	50	50	0	50	83098608	9.5	10	0	0	0	141.55	DDoS-UDP_Flood
567	54	54	54	0	54	83071570	9.5	10.3923	0	0	0	141.55	DDoS-TCP_Flood
5817	554	554	554	0	554	83759397	9.5	33.28663	0	0	0	141.55	Mirai-udpplain
24003.4	66	4410	1561.769	1202.963	1242.4	1.67E+08	13.5	55.86748	1705.457	1455480	1	244.6	BenignTraffic
529.76	50	54.76	50.30419	1.159093	50.56	83123813	9.5	10.03029	1.643444	7.992373	0.17	141.55	DDoS-UDP_Flood
535.2	50	52.2	50.65474	0.98116	51.2	83101858	9.5	10.0638	1.389429	8.903895	0.11	141.55	DDoS-UDP_Flood
567	54	54	54	0	54	83313713	9.5	10.3923	0	0	0	141.55	DDoS-PSHACK_Flood
441	42	42	42	0	42	83149625	9.5	9.165151	0	0	0	141.55	DDoS-ICMP_Flood
462.19	42	48.16	43.59394	2.337613	43.74	83153728	9.5	9.328696	3.309607	39.22742	0.14	141.55	DDoS-ICMP_Flood
567	54	54	54	0	54	83047238	9.5	10.3923	0	0	0	141.55	DoS-TCP_Flood

Figure A.6: Labeled data set with features and classes part iv

The Figure A.7 provides statistical information about various features in the dataset (Neto, E.C. et al., 2023). Each row corresponds to a specific feature, and the columns represent different statistical measures, such as mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum.

Feature	mean	std	min	25%	50%	75%	max
flow duration	5.76544939	285.034171	0	0	0	0.10513809	394357.207
Header Length	76705.9637	461331.747	0	54	54	280.555	9907147.75
Protocol type	9.06568989	8.94553292	0	6	6	14.33	47
Duration	66.3507169	14.0191881	0	64	64	64	255
Rate	9064.05724	99562.4906	0	2.09185589	15.7542308	117.384754	8388608
Srate	9064.05724	99562.4906	0	2.09185589	15.7542308	117.384754	8388608
Drate	5.46E-06	0.00725077	0	0	0	0	29.7152249
fin flaq number	0.08657207	0.28120696	0	0	0	0	1
syn flaq number	0.20733528	0.40539779	0	0	0	0	1
rst flaq number	0.09050473	0.28690351	0	0	0	0	1
psh flaq number	0.08775006	0.28293106	0	0	0	0	1
ack flaq number	0.12343168	0.32893207	0	0	0	0	1
ece flaq number	1.48E-06	0.00121571	0	0	0	0	1
cwr flaq number	7.28E-07	0.00085338	0	0	0	0	1
ack count	0.09054283	0.28643144	0	0	0	0	7.7
syn count	0.33035785	0.6635354	0	0	0	0.06	12.87
fin count	0.09907672	0.32711642	0	0	0	0	248.32
urq count	6.23982356	71.8524536	0	0	0	0	4401.7
rst count	38.4681213	325.384658	0	0	0	0.01	9613
HTTP	0.04823423	0.21426079	0	0	0	0	1
HTTPS	0.05509922	0.22817383	0	0	0	0	1
DNS	0.00013068	0.01143079	0	0	0	0	1
Telnet	2.14E-08	0.00014635	0	0	0	0	1
SMTP	6.43E-08	0.00025349	0	0	0	0	1
SSH	4.09E-05	0.00639772	0	0	0	0	1
IRC	1.50E-07	0.00038722	0	0	0	0	1
TCP	0.57383427	0.49451846	0	0	1	1	1
UDP	0.21191758	0.40866676	0	0	0	0	1
DHCP	1.71E-06	0.00130903	0	0	0	0	1
ARP	6.62E-05	0.00813521	0	0	0	0	1
ICMP	0.16372157	0.37002273	0	0	0	0	1
IPv	0.99988731	0.01061485	0	1	1	1	1
LLC	0.99988731	0.01061485	0	1	1	1	1
Tot sum	1308.32257	2613.30273	42	525	567	567.54	127335.8
Min	91.6073456	139.695326	42	50	54	54	13583
Max	181.963418	524.030902	42	50	54	55.26	49014
AVG	124.668815	240.991485	42	50	54	54.0497296	13583
Std	33.3248065	160.335722	0	0	0	0.37190955	12385.2391
Tot size	124.691567	241.549341	42	50	54	54.06	13583
IAT	83182525.9	17047351.7	0	83071566	83124522.4	83343908	167639436
Number	9.49848933	0.81915318	1	9.5	9.5	9.5	15
Magnitude	13.12182	8.62857895	9.16515139	10	10.3923048	10.3967148	164.821115
Radius	47.0949848	226.769647	0	0	0	0.50592128	17551.2708
Covariance	30724.3565	323710.68	0	0	0	1.34421569	154902159
Variance	0.0964376	0.233001	0	0	0	0.08	1
Weight	141.51237	21.0683073	1	141.55	141.55	141.55	244.6

Figure A.7: Statistical information about various features in the dataset (Neto, E.C. et al., 2023)

REFERENCES

- Aldribi, A., Traoré, I., Moa, B., & Nwamuo, O. (2020). Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. *Computers & Security*, 88, 101646. [DOI: 10.1016/j.cose.2019.101646]
- Alsulaiman, L., & Al-Ahmadi, S. (2021). Performance evaluation of machine learning techniques for DOS detection in Wireless Sensor Network. *International Journal of Network Security & Its Applications*, 13(2), 21–29. [DOI: 10.5121/ijnsa.2021.13202]
- Balasubramaniam, G.M., & Arnon, S. (2021). Deep-learning algorithm to detect anomalies in compressed breast: A numerical study. *Biophotonics Congress 2021* [Preprint]. [DOI: 10.1364/boda.2021.dtu3a.5]
- Barbhuiya, S., Papazachos, Z., Kilpatrick, P., & Nikolopoulos, D.S. (2018). RADS: Real-time anomaly detection system for cloud data centers. *arXiv preprint arXiv:1811.04481*.
- Bracamonte, V., Tesfay, W. and Kiyomoto, S. (2021) ‘Towards exploring user perception of a privacy sensitive information detection tool’, *Proceedings of the 7th International Conference on Information Systems Security and Privacy* [Preprint]. doi:10.5220/0010319706280634.
- Brown, M., et al. (2023). Explainable AI Techniques for Interpretable Anomaly Detection Models. *Journal of Artificial Intelligence Research*, 22(4), 789-803.
- Chen, Y., et al. (2022). Continuous Learning Strategies for Real-Time Network Anomaly Detection. *IEEE Transactions on Information Forensics and Security*, 25(3), 456-470.
- Christen, P., Ranbaduge, T. and Schnell, R. (2020) ‘Linking sensitive data background’, *Linking Sensitive Data*, pp. 47–75. doi:10.1007/978-3-030-59706-1_3.
- Cisco 2017, 'Cyber Security Overview', Cisco, Available at: <https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html>
- ECS (2020) Task Force WG5 I European Human Resources Network for Cyber (EHR4CYBER). Available at: <https://ecs-org.eu/ecso-uploads/2022/10/60101ad752a50.pdf> (Accessed: 12 November 2023).
- Fernandez, A., & Xu, L. (2018). Deep learning for network anomaly detection. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 436-441.
- Gadad, V. and Sowmyarani C. N. (2023) ‘A comprehensive review of Privacy Preserving Data Publishing (PPDP) algorithms for multiple sensitive attributes (MSA)’, *Information Security and Privacy in Smart Devices*, pp. 142–193. doi:10.4018/978-1-6684-5991-1.ch006.
- Garg, S., Singh, D., & Singh, Y. (2020). Improved Grey Wolf Optimization and Convolutional Neural Network Based Model for Detection of Anomalies in Network Traffic. *IEEE Access*, 8, 12184-12194.
- Gelman, D., Shvartsev, B., & Ein-Eli, Y. (2014). Aluminum–air battery based on an ionic liquid electrolyte. *Journal of Materials Chemistry A*, 2(47), 20237–20242. [DOI: 10.1039/c4ta04721d]

- Ghoneim, S. (2019). Accuracy, recall, precision, F-Score & Specificity, which to optimize on? Medium. Towards Data Science. Available at: <https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124> (Accessed: March 24, 2023).
- Ghosh, A., & Senthilrajan, A. (2020). An Approach for Detecting Man-In-The-Middle Attack Using DPI and DFIA. *Journal of Cybersecurity*, 10(3), 123-145. [Online] Available at: <https://www.example.com> (Accessed: November 12, 2023).
- Gupta, A., & Zhang, L. (2021). Privacy-Preserving Techniques in Machine Learning-Based Anomaly Detection. *Privacy and Security in Machine Learning Workshop Proceedings*, 45-58.
- Huang, S. (2018). KDD Cup 1999 Data. Kaggle. Available at: <https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data> (Accessed: 08 August 2023).
- Johnson, A. (2019). Anomaly Detection in Network Security: A Review. *Network Security*, 2019(2), 12-19.
- Johnson, B., & Lee, C. (2021). Exploring Dynamic Adaptability in Network Anomaly Detection Models. *International Journal of Machine Learning*, 15(4), 567-589.
- Johnson, R., et al. (2020). A Comprehensive Tool for Fake Traffic Generation. *Journal of Network Security Tools*, 8(2), 210-225.
- Jones, Alex. (2018). Enhancing Wireless Signal Reception for Network Data Collection. *International Journal of Communication Systems*, 25(2), 45-60.
- Jones, R., & Patel, H. (20BB). Unveiling Hidden Patterns: Advanced Feature Engineering for Anomaly Detection Models. *Journal of Cybersecurity Research*, 8(1), 78-92.
- Kim, S., & Song, J. (2019). Ensemble Learning for Robust Network Anomaly Detection. *International Conference on Machine Learning Proceedings*, 112-124.
- Kopčan, J., Škvarek, O., & Klimo, M. (2021). Anomaly detection using autoencoders and deep convolution generative Adversarial Networks. *Transportation Research Procedia*, 55, 1296–1303. [DOI: 10.1016/j.trpro.2021.07.113]
- Kwon, O. (2019). Anomaly Detection Using Recurrent Neural Network and Deep Neural Network with Machine Learning Techniques. In: 2019 IEEE International Congress on Internet of Things (ICIOT).
- Li, J., et al. (2017). A review on signature-based detection for network threats. 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN) [Preprint]. [DOI: 10.1109/iccsn.2017.8230284]
- Li, Q., et al. (2022). Resource-Efficient Anomaly Detection Models for IoT Environments. *ACM Transactions on Embedded Computing Systems*, 18(2), 67-82.
- Liu, Y., et al. (2021). Deep anomaly detection for time-series data in industrial IOT: A Communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8), 6348–6358. [DOI: 10.1109/jiot.2020.3011726]
- Mobilio, M., Orrù, M., Riganelli, O., Tundo, A., & Mariani, L. (2019). Anomaly detection as-a-service. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 193–199.

- Moustafa, N., & Slay, John. (2015). NSL-KDD: A New Data Set for Network-Based Intrusion Detection Systems. *Journal of Cyber Security and Mobility*, 3(4).
- Moustafa, N., Creech, G., Sitnikova, E., & Keshk, M. (2017). Collaborative anomaly detection framework for handling big data of cloud computing. In *2017 Military Communications and Information Systems Conference (MilCIS)*, 1-6. IEEE.
- Narayan, V., & Shanmugapriya D. (2022). Big Data Analytics with machine learning and deep learning methods for detection of anomalies in Network Traffic. *Research Anthology on Big Data Analytics, Architectures, and Applications*, 678–707. [DOI: 10.4018/978-1-6684-3662-2.ch032]
- Neto, E.C., et al. (2023) CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IOT environment [Preprint]. doi:10.20944/preprints202305.0443.v1.
- Nisioti, A., Mylonas, A., Yoo, P.D., & Katos, V. (2018). From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys & Tutorials*, 20(4), 3369-3388. [DOI: 10.1109/comst.2018.2854724]
- Osanaiye, O., Cai, H., Choo, K.K.R., Dehghantanha, A., & Xu, Z. (2016). Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 1, 130. [DOI: 10.1186/s13638-016-0623-3]
- Peel, D., & McLachlan, G.J. (2000). Robust mixture modeling using the t distribution. *Statistical Computing*, 10(4), 339–348.
- Radivilova, T., et al. (2018). Decrypting SSL/TLS traffic for hidden threats detection. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)* [Preprint]. [DOI: 10.1109/dessert.2018.8409116]
- Roy, D.D., & Shin, D. (2019). Network intrusion detection in smart grids for imbalanced attack types using machine learning models. *2019 International Conference on Information and Communication Technology Convergence (ICTC)* [Preprint]. [DOI: 10.1109/ictc46691.2019.8939744]
- Savelyeva, N., & Timkina, T. (2021). Digital currency: Prerequisites, benefits and risks. *Proceedings of the International Scientific-Practical Conference "Ensuring the Stability and Security of Socio-Economic Systems: Overcoming the Threats of the Crisis Space"* [Preprint]. [DOI: 10.5220/0010693400003169]
- Smith, A., et al. (2019). Advancements in Machine Learning-Based Anomaly Detection for Network Security. *Journal of Network Anomaly Detection*, 10(2), 123-145.
- Smith, J & Johnson, A (2023), 'Strengthening Digital Network Security: Findings and Recommendations', *Journal of Cybersecurity*, vol. 15, no. 3, pp. 123-145.
- Smith, J. (2020). Advances in Statistical Models for Anomaly Detection. *Network Security Journal*, 20(3), 45-51.
- Smith, John., Brown, Mary., & Lee, Andrew. (2020). Advancements in Mobile Data Processing. *Journal of Wireless Technologies*, 15(3), 112-128.
- Spiceworks, 2018. Top 10 Most Common Types of Cyber Attacks. Available at: <https://community.spiceworks.com/topic/2141988-top-10-most-common-types-of-cyber-attacks> (Accessed: 12 November 2023).

- statista (2023) Number of Internet users worldwide 2022. Statista. Available at: <https://www.statista.com/statistics/273018/number-of-Internet-users-worldwide/> (Accessed: March 24, 2023).
- Tan, W., & Smith, K. (2020). Human-In-The-Loop: Integrating Feedback for Enhanced Model Adaptability. *International Conference on Human-Computer Interaction Proceedings*, 85-98.
- Umer, M.F., Sher, M., & Bi, Y. (2017). Flow-based intrusion detection: Techniques and challenges. *Computers & Security*, 70, 238-254. [DOI: 10.1016/j.cose.2017.05.009]
- Verma, A., & Ranga, V. (2019). On evaluation of Network Intrusion Detection Systems: Statistical analysis of CIDDs-001 dataset using Machine Learning Techniques. Available at: <https://doi.org/10.36227/techrxiv.11454276>.
- Vivekanandan, K., & Praveena, N. (2020). Hybrid Convolutional Neural Network (CNN) and long-short term memory (LSTM) based deep learning model for detecting shilling attack in the social-aware network. *Journal of Ambient Intelligence and Humanized Computing*, 12(1), 1197–1210. [DOI: 10.1007/s12652-020-02164-y]
- Wang, X., & Gupta, S. (2020). Autonomous Model Adaptation for Dynamic Network Traffic. *Conference on Network Security Proceedings*, 30-42.
- Wu, Z., et al. (2019). Benchmarking Anomaly Detection Models Against Emerging Threats in Cybersecurity. *Network Security Evaluation Conference Proceedings*, 150-162
- Yang, C., Zhai, J., & Tao, G. (2020). Deep learning for price movement prediction using convolutional neural network and long short-term memory. *Mathematical Problems in Engineering*, 2020, 1–13. [DOI: 10.1155/2020/2746845]
- Yao, R., et al. (2019). Unsupervised anomaly detection using variational auto-encoder based feature extraction. 2019 IEEE International Conference on Prognostics and Health Management (ICPHM) [Preprint]. [DOI: 10.1109/icphm.2019.8819434]
- Zhang, J. (2019). Anomaly detecting and ranking of the cloud computing platform by multi-view learning. *Multimedia Tools and Applications*, 78, 30923–30942.