



School of Engineering and Informatics

CYBER THREAT DETECTION USING MACHINE LEARNING

Course : MSC Computer Science (Conversion)


Submission date: 27.08.2024

Candidate Number: 284500

Supervisor: Daniel Creed

Statement of Originality and Intellectual Property Rights

This report is submitted as part requirement for the degree of MSc in Computer Science at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signed: _____

Date: 27 August 2024

ACKNOWLEDGEMENTS

My sincere gratitude is extended to Professor Daniel Creed for his important counsel, encouragement, and support during this research. Your insightful feedback and unwavering commitment to my learning have been instrumental in shaping the direction and success of this work. I am truly grateful for the time you took to help me navigate complex concepts and for always pushing me to think critically and strive for excellence. This project has been a significant learning experience, and I owe much of that to your mentorship. Thank you for being a constant source of inspiration and for believing in my potential.

ABSTRACT

In recent years, the rapid proliferation of cyber threats has necessitated the development of advanced methods for effective detection and mitigation. This study focuses on detecting botnet activities using machine learning techniques, because botnet-related issues can lead to numerous unwanted activities and scams, specifically leveraging the CTU-13 dataset, which includes real network traffic data capturing various botnet scenarios. This study's main objective is to evaluate how well different machine learning models detect cyberthreats. Specifically, the study will examine the efficacy of Support Vector Machines (SVM), Long Short-Term Memory (LSTM), Random Forest, Logistic Regression, and a hybrid model that combines these techniques to increase detection accuracy.

This paper describes the development process and tools for a botnet detection system. This research makes use of important libraries including Scikit-Learn, Pandas, Theano, Matplotlib, Pickle, and NumPy. The study describes the use of these tools and highlights the feature extraction methods that need significant domain expertise and human labour. A packet's size, bytes, length, source and destination addresses, and associated protocols are among the most important features that are extracted for examination. The protocols that Internet Relay Chat (IRC), DNS, and P2P use—and that the Command and Control (C&C) servers use to exploit them—are used to categorise botnets.

.

This study underscores the potential of integrating multiple machine learning techniques to create a more effective solution for detecting cyber threats compared to traditional methods.

Table of Contents

Acknowledgements.....	2
Abstract.....	4
Chapter 1: Introduction.....	7
1.1 Aims and Objectives.....	7
1.2 Research Motivation.....	8
1.3 Research Questions.....	9
1.4 Research Outcomes.....	10
Chapter 2: Literature Review.....	11
2.1 Introduction to Cyber Threat Detection.....	11
2.1.1 Cyber-Threats and Cyber-Attacks.....	11
2.1.2 Types of Cyber-Attacks.....	12
2.2 Machine Learning in Cybersecurity.....	12
2.2.1 Taxonomy of Machine Learning Algorithms.....	13
2.2.2 Machine Learning for Threat Detection.....	15
2.2.3 Supervised Learning.....	16
2.2.4 Unsupervised Learning.....	17
2.2.5 Deep Learning.....	17
2.3 Botnet as a Cyber Threat.....	18
2.3.1 Why Botnets Were Chosen for This Project?.....	18
2.3.2 Botnet Architecture.....	19
2.4 Related Work.....	21
2.5 Dataset.....	22
Chapter 3: Methodology.....	24
3.1 Technology Stack.....	24
3.2 Data Preprocessing.....	27
3.2.1 Data Loading and Initial Exploration.....	27
3.2.2 Calculation of Missing Values.....	28
3.2.3 Feature Selection and Dimensionality Reduction.....	28
3.2.4 Data Normalization/Scaling.....	30
3.2.5 Data Splitting.....	30
3.2.6 Handling Imbalanced Data.....	30
3.2.7 Data Visualization.....	31

3.3 Selected Algorithms.....	33
3.3.1 Support Vector Machine (SVM).....	33
3.3.2 Long Short-Term Memory (LSTM).....	36
3.3.3 Hybrid Model (SVM + LSTM)	40
3.3.4 Logistic Regression.....	44
3.3.5 Random Forest.....	47
3.3.6 Hybrid Model (Random Forest + Logistic Regression)	49
3.4 Evaluation Metrics.....	52
3.5 Model Comparison.....	53
Chapter 4: Discussion, Limitation, Conclusion And Future Studies	56
4.1 Why Random Forest Performs the Best?.....	56
4.2 Limitations of Hybrid Model.....	56
4.3 Conclusion.....	57
4.4 Future Studies.....	58
Chapter 5: Bibliography.....	59

CHAPTER 1: INTRODUCTION

1.1 AIMS AND OBJECTIVES

This research is focused on improving the security of digital infrastructure by advancing the detection and mitigation of cyber threats using machine learning (ML) techniques. The goal is to tackle the practical challenges of integrating ML into cybersecurity, aiming to create scalable and adaptable solutions that effectively address the growing complexity and frequency of cyber-attacks. Traditional cybersecurity methods often struggle to identify new and sophisticated threats, whereas ML offers the capability to analyse large datasets and detect complex patterns, providing a more robust and dynamic approach to threat detection. The primary motivation behind this study is to enhance the effectiveness of cyber threat detection systems and contribute to the field of cybersecurity through the innovative application of ML techniques. Unlike studies that focus on a single ML algorithm or a narrow range of threats, this research will explore a variety of ML techniques and propose hybrid models that combine multiple algorithms. Hybrid models have the potential to leverage the strengths of different algorithms, compensating for individual weaknesses and leading to more accurate and reliable threat detection results.

This study engages with key theoretical concepts related to supervised and unsupervised ML, anomaly detection, hybrid model development, and the integration of ML into real-time cybersecurity systems. By investigating these areas, the research aims to build a comprehensive understanding of how advanced ML methodologies can be effectively employed to strengthen cybersecurity measures.

This study systematically examines various ML algorithms, evaluates their performance in detecting different types of cyber threats, and explores methods to enhance their effectiveness. By proposing and assessing hybrid models that integrate multiple ML algorithms, the research aims to demonstrate how these combinations can achieve better detection accuracy and reliability compared to single-algorithm approaches.

Key aspects of this research include analysing the types of threats detectable through ML, identifying the most relevant performance metrics for evaluating ML models in cybersecurity, and addressing the practical challenges of implementing ML-based solutions in real-world security contexts. Through this comprehensive investigation, the study aims to contribute valuable insights to the field of cybersecurity, particularly in the effective application of machine learning for threat detection and prevention. This research seeks to advance the development

of more resilient and proactive cybersecurity strategies capable of protecting digital infrastructures from increasingly complex and widespread threats.

1.2 RESEARCH MOTIVATION

My interest in this project is deeply personal and driven by a passion that has evolved over time. Despite completing a bachelor's degree in English literature and having a background in biology during my schooling years, I always felt like I was following a path that others had set for me, without truly exploring my own interests. It wasn't until my second year of undergraduate studies that I stumbled upon computer science and cybersecurity—a discovery that ignited a genuine passion within me.

When I first approached my parents about shifting my focus to cybersecurity, they were hesitant and advised me to stick with my current path. However, I couldn't ignore the excitement I felt for this field. Determined to follow my heart, I took matters into my own hands and enrolled in programming classes on my own. I learned C, C++, HTML, Python, and advanced Python, all while juggling my existing studies.

As I delved deeper into programming, I realized that this was the direction I wanted my career to take. I began applying for master's programs abroad in cybersecurity, but my lack of formal experience led to a series of rejections. Still, I persevered, driven by a deep-seated belief that this was where I belonged. After many setbacks, I finally received an offer letter from the University of Sussex—a moment that felt nothing short of miraculous. This acceptance was a turning point for me, as it allowed me to channel my passion into my dissertation, paving the way toward my dream career in cybersecurity.

The journey has been filled with both challenges and triumphs. I remember the fear I felt when I completed my first Python programming project, only to be rewarded with a score of 71%. That achievement, though modest, brought me immense satisfaction and confirmed that I was on the right path. Each step has been a testament to my passion for cybersecurity, a field that continues to fascinate and inspire me.

This project is not just an academic requirement for me; it represents a vital key to unlocking my future in cybersecurity. It's an opportunity to live out a dream that I once thought was out of reach and to pursue a career that genuinely excites me. I am more determined than ever to make this dream a reality, and this project is the next step in that journey.

1.3 RESEARCH QUESTIONS

<i>Q1 How effective are current machine learning algorithms in detecting cyber threats compared to traditional methods?</i>	Current machine learning algorithms, particularly those utilizing anomaly detection and pattern recognition, have demonstrated superior performance in identifying botnet activities, which often involve sophisticated and evolving threats. Unlike traditional signature-based methods, which rely on known patterns, machine learning algorithms can adapt to new and unknown botnet behaviours, providing dynamic detection capabilities that improve over time with exposure to new data.
<i>Q2 What types of botnet behaviours are most successfully identified by machine learning models?</i>	Machine learning models excel at detecting botnet behaviours that exhibit distinct patterns or anomalies, such as unusual network traffic, abnormal communication patterns with command and control (C&C) servers, and unexpected changes in data flow. These models are particularly effective in identifying both centralized botnets (like IRC-based) and decentralized botnets (like peer-to-peer) by monitoring deviations from normal network activity that signal potential botnet-related incidents.
<i>Q3 What are the main challenges in integrating machine learning into existing cybersecurity frameworks for botnet detection?</i>	The integration of machine learning into existing cybersecurity frameworks for botnet detection faces several challenges. These include ensuring the availability of high-quality, diverse datasets to train models effectively, the complexity of real-time data processing, and the need for substantial computational resources. Moreover, model interpretability remains a critical issue, as understanding the decision-making process of ML models is essential for gaining trust and effectively responding to detected threats. Lastly, aligning ML models with existing security protocols

	and ensuring they operate seamlessly within real-time environments is a complex task.
Q4 How can machine learning models be improved to enhance their accuracy and reliability in detecting botnet activities?	Improvements to machine learning models for botnet detection can be achieved by developing hybrid models that combine the strengths of multiple algorithms, such as combining supervised learning with unsupervised anomaly detection methods. Enhanced feature engineering techniques, which focus on extracting and utilizing the most relevant network traffic data, can also improve model performance. Additionally, incorporating transfer learning, where models trained on one type of network data are adapted to other types, can increase model versatility. Continuous updates and retraining with new data, along with feedback loops that allow models to learn from real-time incidents, are essential for maintaining and enhancing their effectiveness over time.

1.4 RESEARCH OUTCOMES

To enhance cyber threat detection by improving the existing detection models and integrating hybrid approaches.

- a.** To use expert evaluations and experimental network traffic data to enhance existing machine learning models (e.g., Random Forest, SVM, LSTM, Logistic Regression, and Random Forest).
- b.** To develop and validate hybrid models that combine multiple machine learning techniques to enhance detection accuracy and robustness.

Specific Outcome a: Focused on enhancing individual machine learning models for cyber threat detection, this objective was achieved through the application of advanced machine learning techniques and comprehensive feature analysis to develop models that are both accurate and explainable. Success indicators included statistical measures such as accuracy,

precision, recall, and F1-score. A critical requirement was ensuring that the models remained transparent and explainable to foster trust and reliability in practical applications.

Specific Outcome b: This objective centered on developing and validating hybrid models that integrate multiple machine learning techniques. By combining models like SVM, LSTM, Logistic Regression, and Random Forest, the goal was to create a more robust detection system. Given the complex nature of cyber threats, this was an exploratory outcome requiring further testing and validation. Success indicators for this objective included improved detection accuracy and robustness compared to individual models.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION TO CYBER THREAT DETECTION

A cyber threat refers to any activity aimed at compromising the security of an information system. This includes attempts to alter the system's availability, integrity, or confidentiality, as well as efforts to disrupt digital activities overall. The **cyber threat environment** encompasses the online domain where malicious actors carry out cyber threats. This environment includes all networks, devices, and processes connected to the Internet that could be targeted by cyber attackers.

2.1.1 CYBER-THREATS AND CYBER-ATTACKS

In the current cybersecurity landscape, the terms "cyber-threat" and "cyber-attack" are often used interchangeably, creating ambiguity in their definitions. However, these concepts are critical for understanding hostile activities in cyberspace. The U.S. Government broadly defines cyber-threats as a range of malicious activities in cyberspace, including website defacement, espionage, and denial-of-service attacks. In contrast, the Oxford English Dictionary distinguishes between the two, defining a cyber-threat as the potential for harm, while a cyber-attack refers to the actual execution of those malicious attempts, turning threats into incidents.

2.1.2 TYPES OF CYBER-ATTACKS

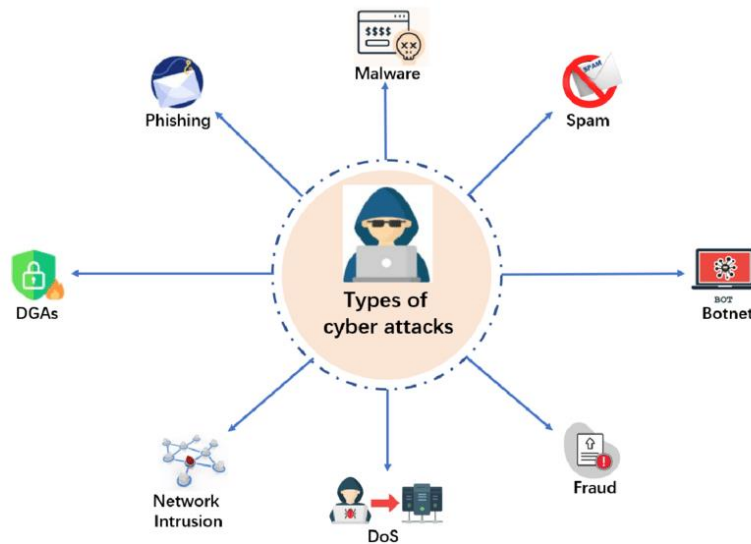


Figure 1: Common Types of Cyber Attacks.

There are many types of cyber threats, including phishing, malware, spam, fraud, and DoS attacks. However, I have chosen to focus on botnets for this report because of a personal fascination with their complexity and impact. Botnets, networks of infected devices controlled by cybercriminals, can execute large-scale attacks like Distributed Denial of Service (DDoS), spam distribution, and data theft. My interest in cybersecurity grew when I realised how these threats disrupt both individuals and organisations, causing substantial harm. By focusing on botnet detection and mitigation, I hope to contribute to creating more secure digital environments.

2.2 MACHINE LEARNING IN CYBERSECURITY

Machine learning (ML) is commonly recognized as a subset of Artificial Intelligence (AI), closely tied to fields such as data mining, computational statistics, analytics, and data science. Its primary focus is on enabling systems to learn from historical data. In the context of my project, machine learning models are developed using a combination of rules, procedures, and complex functions or equations. These models are designed to identify interesting patterns in data, recognize sequences, and predict behaviours. Consequently, ML plays a significant role in enhancing cybersecurity within my project, offering valuable tools for detecting and mitigating threats by leveraging data-driven insights.

2.2.1 TAXONOMY OF MACHINE LEARNING ALGORITHMS

The taxonomy of machine learning models in cybersecurity can be broadly divided into three categories: shallow models, deep learning models, and reinforcement learning.

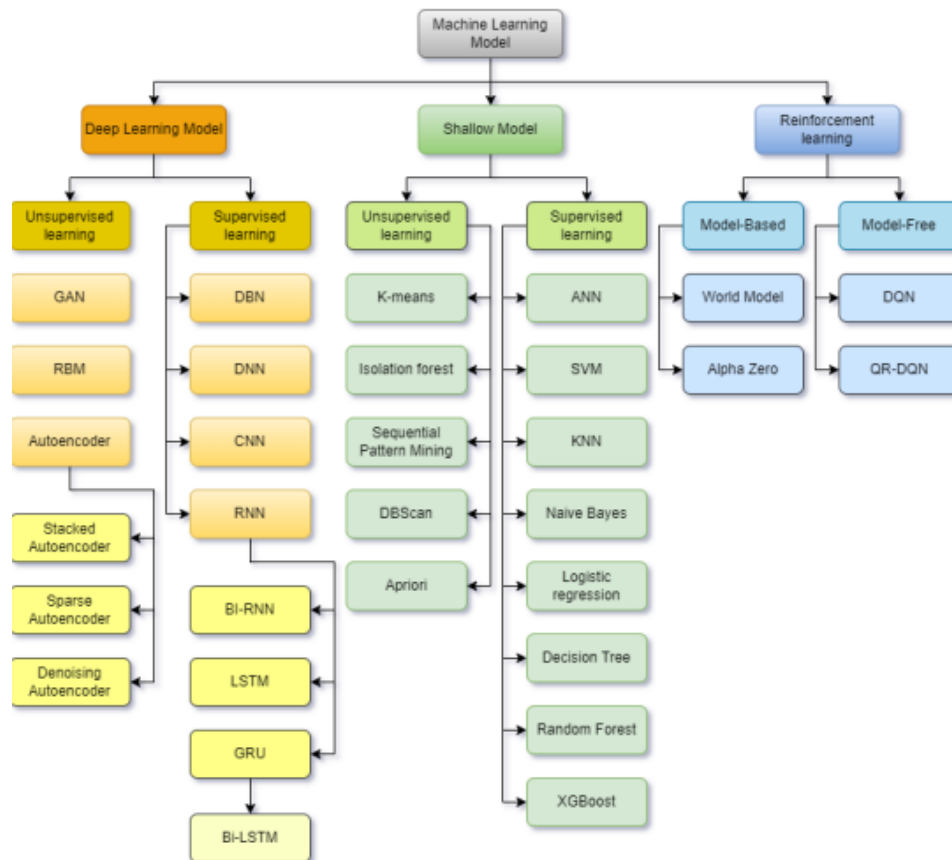


Figure 2: Overview of Machine Learning Models and Techniques.

Shallow Models:

Shallow models, commonly used in traditional machine learning, are further classified into supervised and unsupervised learning. In supervised learning, models operate with labeled data, allowing them to predict outcomes based on past examples. For instance, algorithms like Naïve Bayes use probabilistic distributions to assign data to specific categories, while Decision Trees create a tree-like structure from the training data, enabling the classification of new records based on this structure. Random Forest improves on this by generating multiple decision trees and using a voting mechanism to increase classification accuracy. Support Vector Machines (SVM) create linear decision boundaries for binary classification, with the ability to handle nonlinear datasets through the use of kernel tricks.

Deep Learning Models:

Deep learning models take a different approach, utilizing artificial neural networks to classify or cluster data. Unlike traditional machine learning models, deep learning models are often referred to as "black box" models due to their complex, non-transparent decision-making processes. These models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are designed to identify patterns within data, adapting their internal parameters, or weights, over time through training.

CNNs are particularly effective in image classification tasks and have been adapted to classify cybersecurity datasets by transforming data into image-like formats. RNNs, along with their variants like LSTM (Long Short-Term Memory) and Bi-LSTM, are well-suited for handling temporal data, making them useful in scenarios where time-based sequences are critical.

Reinforcement Learning:

Reinforcement learning represents a unique approach where models learn by interacting with their environment, receiving rewards or penalties based on their actions. This method helps models differentiate between long-term and short-term goals, improving decision-making over time. A notable example is Deep Q Networks (DQN), which utilizes deep learning to map states to actions, reducing the complexity compared to traditional Q-learning. Variants like QR-DQN enhance this approach by using quantile regression to model potential distributions, offering more nuanced predictions.

Application to Cybersecurity:

In the context of my project, these machine learning techniques are applied to develop an effective intrusion detection system (IDS). Shallow models have been extensively researched and are well-suited for tasks such as labeling, efficiently detecting attacks, and managing data. However, deep learning and reinforcement learning models offer advanced capabilities, particularly in handling complex, evolving threats. By leveraging these models, my project aims to create a robust system capable of detecting both known and unknown threats, enhancing the overall security posture of the environment.

2.2.2 MACHINE LEARNING FOR THREAT DETECTION



Figure 3: Comparison of Supervised and Unsupervised Machine Learning Approaches for Threat Detection.

In this project, the security lifecycle involves three key stages: prevention, detection, and response. It is well understood that completely preventing every cyber threat is not feasible, while the response phase is focused on managing and mitigating the damage after an incident occurs. This project utilizes two primary approaches to cyber threat detection: misuse-based detection and anomaly-based detection. Misuse-based detection, often referred to as signature-based detection, involves identifying known patterns associated with specific threats, operating under the assumption that future threats will exhibit similar patterns. On the other hand, anomaly-based detection focuses on establishing a baseline of "normal" behaviours and detecting deviations from this norm, which may indicate new or previously unknown threats. These approaches are complementary in our system: misuse-based detection provides high accuracy for identifying known threats, while anomaly-based detection offers the ability to detect novel attacks, though it may result in more false positives.

Prior to the incorporation of machine learning, these detection methods relied heavily on manually defined rules and patterns, which were not only labor-intensive but also struggled to keep up with the growing complexity of cyber environments. We have integrated ML-based solutions to enhance detection efficiency and effectiveness. Machine learning enables our system to automatically learn and adapt, identifying subtle patterns—so-called "weak" signals—in the data that might be missed by traditional methods, thereby improving detection accuracy.

In applying ML to our threat detection system, we carefully considered the choice between supervised and unsupervised learning methods. Supervised learning is used in scenarios where labeled data is available, allowing our models to learn from past examples to predict future threats. This is particularly useful for well-understood threats, where accurate labeling is possible. Unsupervised learning, which does not require labeled data, is employed in

situations where the data is more complex or where labeling is impractical. This allows our system to identify new and emerging threats without prior knowledge.

In summary, the integration of machine learning has significantly improved our ability to detect both known and unknown threats, enhancing the overall security and resilience of the system.

2.2.3 SUPERVISED LEARNING

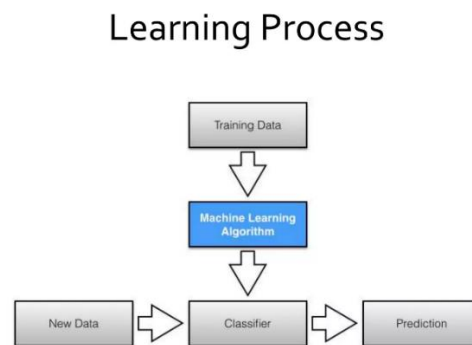


Figure 4 The Machine Learning Process Flow

The machine learning model's development process involves two main stages: training and testing. During the training phase, the model is fed with training data, which includes input samples that the model analyses to detect patterns and build its predictive capabilities. This process is driven by a learning algorithm, which evaluates the data to construct a robust model. Once the model is trained, it is tested on new, unseen data—often called test or production data—to make predictions or classifications. The output at this stage is the predicted labels or classifications.

In this project, supervised learning techniques are primarily used for classification tasks, where the model is trained to recognize and categorize data based on a predefined classification system. The dataset provided to the model contains features and corresponding labels (known outputs). The goal is to develop a model that can accurately predict the correct label for new data based on its features, by minimizing prediction errors during training.

Supervised learning is particularly effective for training models like neural networks and decision trees, which depend on predefined classifications for accurate predictions. It is versatile, applicable to both classification (categorical outputs) and regression (continuous outputs) tasks, making it a powerful tool in cybersecurity for detecting and mitigating various threats.

2.2.4 UNSUPERVISED LEARNING

Unsupervised learning, a branch of artificial intelligence, involves machine learning models that learn from data without the need for human supervision. Unlike supervised learning, which depends on labeled data, unsupervised learning works with unlabelled data, identifying patterns and insights autonomously.

In the context of cybersecurity, unsupervised learning is particularly useful for detecting anomalies and uncovering hidden patterns within large datasets. These models employ self-learning algorithms to analyse data, recognizing similarities, differences, and patterns. They can organize data into clusters or identify features that are essential for categorizing data, making them valuable for discovering previously unknown threats and anomalies.

Although my project primarily focuses on supervised learning techniques, we have explored the potential of integrating unsupervised learning to enhance the system's detection capabilities. While unsupervised learning was not the main focus, its application could provide additional insights into unknown or emerging threats, thereby contributing to a more comprehensive security framework. This underscores the importance of combining both supervised and unsupervised learning methods to build a robust and adaptive cybersecurity system.

2.2.5 DEEP LEARNING

Deep learning, a specialized branch of machine learning, utilizes multilayered neural networks, known as deep neural networks, to mimic the complex decision-making processes of the human brain. In my project, deep learning was crucial for enhancing cybersecurity, specifically in detecting and analysing cyber threats. Unlike traditional machine learning models, which typically employ shallow networks with one or two layers, deep learning models use networks with multiple layers—sometimes hundreds or thousands—allowing them to capture intricate patterns within the data.

A key advantage of deep learning, especially Long Short-Term Memory (LSTM) networks, is their ability to handle sequence prediction tasks effectively. LSTMs are particularly adept at analysing time series data, such as network traffic patterns, making them invaluable for detecting anomalies that might signal cyber threats. By leveraging LSTM networks, my project aimed to identify subtle patterns and irregularities in real-time, significantly improving the system's ability to detect and respond to evolving threats.

This deep learning approach offers a distinct advantage over traditional methods by enabling models to autonomously adapt to new threats without requiring explicit programming. The continuous evaluation and refinement of predictions further enhance the system's effectiveness, making deep learning an essential tool in developing robust, adaptive cybersecurity solutions. My project demonstrates the significant potential of deep learning in advancing cybersecurity strategies.

2.3 BOTNET AS A CYBER THREAT

A botnet, short for "robot network," consists of a network of compromised devices connected to the internet, each running one or more malicious bots. These infected devices are linked to a Command and Control (C&C) server, where they await instructions to carry out harmful activities. Botnets are recognized as a significant cybersecurity threat due to their capability to coordinate large-scale attacks that can disrupt services, steal sensitive information, and cause widespread damage.

Botmaster Control: The botnet is operated by a botmaster, an individual or group that remotely controls the network. The botmaster sends commands to the bots within the network, directing them to perform various illicit activities. These activities may include sending out massive volumes of spam emails or launching Distributed Denial of Service (DDoS) attacks, which can overwhelm and disable targeted systems. The botmaster typically employs propagation techniques to exploit vulnerabilities in computers, thereby converting them into "slaves" or "zombies" under their control.

Understanding the behaviours and mechanisms of botnets has been crucial in developing effective detection and mitigation strategies. By analysing the communication patterns and operational tactics of botnets, the project aims to enhance the ability to detect and neutralize these threats before they can cause significant harm.

2.3.1 WHY BOTNETS WERE CHOSEN FOR THIS PROJECT?

The decision to concentrate on botnets in this project is motivated by their widespread and evolving presence as a cyber threat. Botnets present a significant challenge to cybersecurity because they can remain concealed within normal network traffic and have the potential to inflict extensive damage. By focusing on botnets, this project seeks to create and implement robust detection and mitigation techniques to defend against these threats. This focus on botnets aims to tackle a vital issue in cybersecurity by utilising advanced technologies to

develop effective strategies for detection and mitigation. The project is intended to contribute to academic research and offer practical solutions to strengthen cybersecurity defences against botnets.

2.3.2 BOTNET ARCHITECTURE

In my project, the focus was on analysing and detecting centralized botnets using advanced machine learning techniques. Centralized botnets are particularly concerning in cybersecurity due to their organized structure and potential to cause widespread damage. This type of botnet architecture is characterized by a central Command and Control (C&C) server, which the botmaster uses to manage and coordinate all the infected bots within the network.

Centralized Architecture:

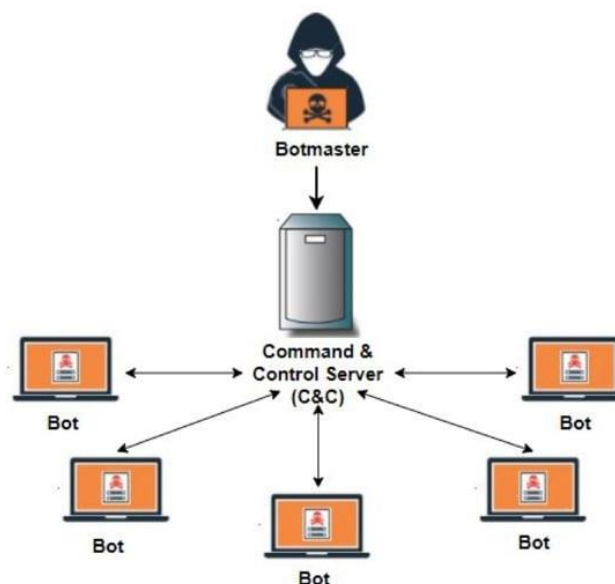


Figure 5: Botnet Architecture: Command and Control (C&C) Structure.

In my project, the decision to focus on centralized botnets was driven by the unique challenges and opportunities they present in the realm of cybersecurity. The centralization of control in these botnets makes them both a potent threat and a potential point of weakness. By targeting the C&C server, it is possible to disrupt the botnet's operations entirely, making detection and mitigation strategies highly effective.

Protocols in Centralized Botnets:

Two common protocols used in centralized botnet architectures are Internet Relay Chat (IRC) and Hypertext Transfer Protocol (HTTP). Both protocols have distinct characteristics that make them suitable for C&C communication:

IRC-Based Botnets:

A popular protocol for synchronous online conferences and real-time text messaging is called Internet Relay Chat (IRC). For botnets, it has emerged as one of the most widely used C&C channels because of a number of significant benefits:

- ✓ **Real-Time Communication:** IRC allows the botmaster to communicate with the bots instantaneously, enabling quick coordination and response.
- ✓ **Support for Large Numbers of Clients:** IRC can manage a vast number of bots simultaneously, making it scalable for large botnet operations.
- ✓ **Versatile Network Topologies:** The protocol works well with various network configurations, allowing the botnet to be flexible in its deployment.
- ✓ **Open-Source and Expandable Design:** IRC's open-source nature allows the botmaster to customize and expand the botnet as needed.

Some of the most infamous IRC-based botnets include Spybot, Agobot, SDBot, and GT Bot. In my project, the focus on IRC-based botnets was chosen because of their prevalence and the specific challenges they present in detection. By leveraging machine learning techniques, the project aimed to develop robust methods for identifying the communication patterns and operational signatures of these botnets, ultimately enhancing the effectiveness of cybersecurity defenses.

The decision to analyse centralized botnets, particularly those using IRC, was integral to the project's goals. The centralized nature of these botnets presents both a significant threat and a key opportunity for disruption. By focusing on this architecture, the project sought to create targeted solutions that could neutralize these botnets by exploiting their reliance on a central C&C server. This approach not only highlights the potential vulnerabilities within centralized botnets but also demonstrates the power of machine learning in improving cybersecurity measures.

Botnet Threats

Compared to more conventional malware like viruses and worms, botnets represent a bigger hazard. Numerous botnet attack types, including as click fraud, distributed denial-of-service (DDoS) attacks, spam campaigns, cyberwarfare, resource exploitation, and the theft of private data, have been identified by the HoneyNet project. Furthermore, research suggests that a wide variety of illegal operations and cybercrimes can be committed using botnets.

2.4 Related Work

The literature on botnet detection extensively covers network flow-based techniques within traditional IP networks. These approaches can be broadly categorized into those focused on detecting specific classes of botnets and those aimed at identifying generic botnets, regardless of their underlying Command and Control (C&C) communication network structure. Flow-based detection methods rely on extracting statistical features from the network flows within a monitored network, which are then used to uniquely identify botnet-related activities.

Llivadas et al. (2006) introduced the application of machine learning algorithms for botnet detection, marking a significant advancement in this field. Their approach leveraged machine learning to automatically detect botnets using a set of statistical features derived from network flows. Both supervised and unsupervised learning algorithms have since been applied to various state-of-the-art flow-based botnet detection methods.

However, the emerging architecture of Software-Defined Networks (SDNs) has not been extensively explored by botnet researchers. The few existing approaches typically focus on determining whether a detected flow belongs to a botnet, without providing deeper analysis. In contrast, my project applies a multiclass supervised machine learning approach to detect botnet-related flows specifically within SDNs, addressing the gap in current research.

.

To address the need for near real-time detection, the work by David Zhao (2012) proposed a generic botnet detection approach based on interval-based analysis. This method involved dividing the flow dataset into time-based intervals and analysing one interval at a time, with the **REPTree** machine learning algorithm being used to extract features indicative of botnet activity. The study found that a 300-second interval provided the best results for detecting botnets in traditional IP networks.

My project builds on this body of work by applying advanced machine learning techniques to detect botnets within SDNs, aiming to improve both the detection accuracy and the ability to identify novel botnet threats in dynamic network environments.

2.5 DATASET

For this project, I selected the CTU-13 dataset, which is renowned for its comprehensive coverage of botnet activity captured in a real network environment. The dataset, collected in 2011 at CTU University in the Czech Republic, consists of 13 distinct scenarios, each capturing different malware behaviours and interactions within a network. This dataset was chosen due to its rich diversity and extensive labelling, making it one of the most detailed and widely used datasets in botnet research.

The CTU-13 dataset includes three categories of network traffic: Botnet, Normal, and Background. Botnet traffic originates from hosts infected with malware, Normal traffic is generated by verified, non-infected hosts, and Background traffic includes all other network activity where the exact nature is uncertain. The detailed labelling provided on a flow-by-flow basis across these categories was a crucial factor in selecting this dataset for my project, as it enables precise training and evaluation of machine learning models.

I chose the CTU-13 dataset because it offers a substantial and realistic record of botnet traffic intermixed with normal and background traffic, which is essential for developing and testing robust machine learning models aimed at detecting and analysing botnet activities. The variety of botnet scenarios, protocols used, and actions performed by the malware in this dataset provided a comprehensive foundation for the project, allowing for the creation of models that can generalize well across different types of botnet threats.

Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)										
Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		
5		✓		✓					✓	UDP and ICMP DDoS.
6				✓						Scan web proxies.
7				✓					✓	Proprietary C&C. RDP.
8				✓						Chinese hosts.
9	✓	✓	✓	✓						Proprietary C&C. Net-BIOS, STUN.
10	✓				✓			✓		UDP DDoS.
11	✓				✓			✓		ICMP DDoS.
12							✓			Synchronization.
13		✓		✓					✓	Captcha. Web mail.

Figure 6: Botnet Scenario Characteristics

The CTU-13 dataset, captured in pcap files, was processed to extract NetFlows and WebLogs for analysis. Initially, unidirectional NetFlows were used, but bidirectional NetFlows later proved more effective, offering better client-server distinction and detailed labels. This project utilized bidirectional NetFlows to build robust machine learning models for botnet detection. The dataset's detailed scenarios, including packet counts, NetFlows, and malware types, were crucial in developing these models.

Scen.	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2,824,636	39,933(1.41%)	30,387(1.07%)	1,026(0.03%)	2,753,290(97.47%)
2	1,808,122	18,839(1.04%)	9,120(0.5%)	2,102(0.11%)	1,778,061(98.33%)
3	4,710,638	26,759(0.56%)	116,887(2.48%)	63(0.001%)	4,566,929(96.94%)
4	1,121,076	1,719(0.15%)	25,268(2.25%)	49(0.004%)	1,094,040(97.58%)
5	129,832	695(0.53%)	4,679(3.6%)	206(1.15%)	124,252(95.7%)
6	558,919	4,431(0.79%)	7,494(1.34%)	199(0.03%)	546,795(97.83%)
7	114,077	37(0.03%)	1,677(1.47%)	26(0.02%)	112,337(98.47%)
8	2,954,230	5,052(0.17%)	72,822(2.46%)	1,074(2.4%)	2,875,282(97.32%)
9	2,753,884	179,880(6.5%)	43,340(1.57%)	5,099(0.18%)	2,525,565(91.7%)
10	1,309,791	106,315(8.11%)	15,847(1.2%)	37(0.002%)	1,187,592(90.67%)
11	107,251	8,161(7.6%)	2,718(2.53%)	3(0.002%)	96,369(89.85%)
12	325,471	2,143(0.65%)	7,628(2.34%)	25(0.007%)	315,675(96.99%)
13	1,925,149	38,791(2.01%)	31,939(1.65%)	1,202(0.06%)	1,853,217(96.26%)

Figure 7: Distribution of Network Flows Across Botnet Scenarios

Flow Type Distribution

This table delves into the distribution of different types of network flows within each scenario:

- **Total Flows:** The total number of network flows recorded for each scenario.
- **Botnet Flows:** The number and percentage of flows originating from infected (botnet) hosts.
- **Normal Flows:** The number and percentage of flows generated by non-infected, normal hosts.
- **C&C Flows:** Command and Control flows, representing communication between the bots and their C&C server.
- **Background Flows:** All other flows that do not fall into the categories of botnet, normal, or C&C flows, representing miscellaneous or uncertain traffic.

The detailed statistics provided in these tables were instrumental in the analysis and processing phases of the project. understanding the distribution of botnet, normal, and background flows allowed for the effective training and validation of machine learning models.

specifically, the focus on scenarios with varied durations, flow counts, and botnet activities ensured that the models could generalize well across different types of botnets behaviours and network environments.

The use of bidirectional netflows, as highlighted in the earlier discussion, was crucial in accurately capturing and analysing these flows, particularly in distinguishing between client-server roles and enhancing the detail of the labels. this comprehensive data allowed for the development of more robust detection algorithms capable of identifying botnet activities in real-time, even in complex and dynamic network environments.

overall, the ctu-13 dataset's richness and the detailed breakdown provided by these tables were critical in advancing the project's goals of improving botnet detection methodologies using machine learning techniques.

CHAPTER 3: METHODOLOGY

3.1 TECHNOLOGY STACK

The following section outlines the technology stack utilized in this project, detailing each component and the rationale behind its inclusion.

Python

Python serves as the foundational programming language for this project due to its extensive ecosystem and ease of use. Python is well-suited for data science and machine learning tasks because of its readability and the availability of powerful libraries that streamline the development process. Given its ability to manage memory automatically and support multiple programming paradigms, Python was the obvious choice for implementing complex machine learning models and handling large datasets like CTU-13.

Jupyter Notebook

Jupyter Notebook was employed as the primary development environment for this project. It is possible to create and share documents with live code, equations, visualisations, and narrative text with this open-source web application. The ability to integrate code execution with data visualization and explanatory text makes Jupyter an ideal tool for documenting the iterative process of data analysis and model development.

Pandas

Pandas is a data manipulation and analysis library in Python, extensively used in this project for loading, cleaning, and preprocessing the CTU-13 dataset. Pandas' powerful data structures, such as DataFrames, enable efficient handling of large datasets, which was essential for merging and processing the different traffic types within the CTU-13 dataset.

NumPy

NumPy is fundamental to scientific computing in Python and was used in this project for numerical operations, including matrix and array operations. It serves as the backbone for data preprocessing tasks, ensuring that data is formatted correctly before being fed into machine learning models.

Scikit-learn.

Scikit-learn, a versatile machine learning library, played a critical role in the data preprocessing and model training phases of this project. It was used for various tasks, including:

- **Standardization:** Utilizing StandardScaler to normalize feature values.
- **Data Splitting:** Employing train_test_split to create training, validation, and test sets.
- **Model Implementation:** Scikit-learn provided essential tools for implementing Support Vector Machines (SVM) and evaluating model performance.

TensorFlow and Keras

TensorFlow, along with Keras (its high-level API), was utilized for developing and training deep learning models in the project. Specifically, Keras was used to build Long Short-Term Memory (LSTM) networks for detecting sequential patterns in network traffic data, which are indicative of botnet activities. The combination of TensorFlow's robust backend and Keras's user-friendly API made it possible to experiment with and fine-tune deep learning models effectively.

SMOTE (Synthetic Minority Over-sampling Technique)

To address the class imbalance in the dataset, SMOTE was applied to oversample the minority class. This technique generated synthetic samples, which balanced the dataset and improved the performance of the machine learning models by allowing them to better learn from the underrepresented class.

seaborn

In this project, **Seaborn** was utilized for creating advanced statistical visualizations that helped in understanding the underlying patterns within the dataset. Seaborn was particularly chosen for its ability to easily generate aesthetically pleasing and informative plots with minimal code. Key uses of Seaborn in this project included:

- **Correlation Heatmaps:** Seaborn was used to generate correlation heatmaps, which were crucial in identifying the relationships between different features in the dataset. This visualization helped in selecting the most relevant features for the machine learning models.
- **Distribution Plots:** Seaborn's capability to plot distributions was leveraged to analyse the distribution of botnet, normal, and background traffic within the dataset. This helped in understanding the spread and concentration of data points across different classes.
- **Boxplots and Violin Plots:** These plots were employed to visualize the distribution and variability of data across different categories. Seaborn made it easy to compare these distributions, which was important for diagnosing the performance of the models and understanding the data characteristics.

Matplotlib

Matplotlib served as the foundational visualization tool in this project, complementing Seaborn by offering more control and customization for plots. It was used extensively for:

- **Confusion Matrices:** Matplotlib was used to create confusion matrices that visualized the performance of the machine learning models, showing the number of correct and incorrect predictions. This was critical for evaluating the accuracy and reliability of the models.
- **Actual vs. Predicted Plots:** The plots comparing actual versus predicted values were generated using Matplotlib. These visualizations were key in assessing how well the models were performing and identifying any potential areas for improvement.
- **Customization:** Matplotlib's flexibility allowed for detailed customization of plots, including setting titles, labels, legends, and color schemes. This was essential for creating clear and professional-quality visualizations that effectively communicated the results of the analysis.

The technologies and libraries selected for this project were chosen for their robustness, flexibility, and ease of integration. By leveraging these tools, the project was able to build, train, and evaluate effective machine learning models for detecting botnet activities in network traffic. The use of Python and its associated libraries ensured that the entire process, from data preprocessing to model deployment, was both efficient and scalable.

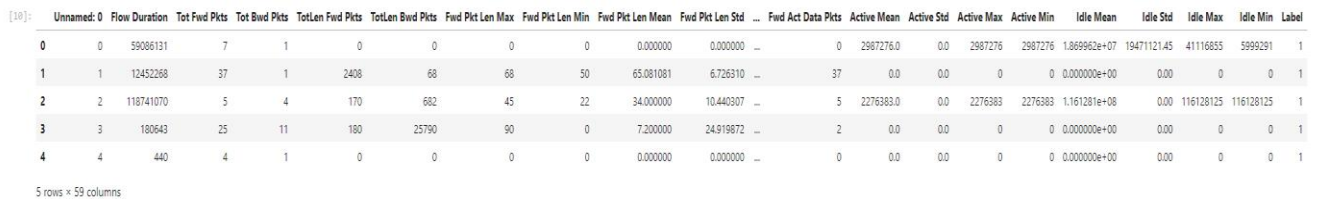
3.2 DATA PREPROCESSING

3.2.1 DATA LOADING AND INITIAL EXPLORATION

The CTU-13 dataset was utilized, specifically focusing on two CSV files: CTU13_Attack_Traffic.csv and CTU13_Normal_Traffic.csv. The first step involved loading these datasets into Pandas DataFrames using the `pd.read_csv()` function. After loading both datasets, they were combined into a single DataFrame using the `pd.concat()` function to facilitate the analysis and processing of all the relevant data together. The combined dataset was then saved to a new CSV file, FINAL YEAR PROJECT.csv, for further use.

Initial Inspection

Once the data was loaded and combined, an initial exploration was conducted to understand its structure and identify any potential issues. The `head()` function was used to display the first five rows of the dataset. This provided an overview of the columns and the kind of data each column contain.



	Unnamed: 0	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std	...	Fwd Act Data Pkts	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	0	59086131	7	1	0	0	0	0	0.000000	0.000000	...	0	2987276.0	0.0	2987276	2987276	1.869962e+07	19471121.45	41116855	5999291	1
1	1	12452268	37	1	2408	68	68	50	65.081081	6.726310	...	37	0.0	0.0	0	0	0.000000e+00	0.00	0	0	1
2	2	118741070	5	4	170	682	45	22	34.000000	10.440307	...	5	2276383.0	0.0	2276383	2276383	1.161281e+08	0.00	116128125	116128125	1
3	3	180643	25	11	180	25790	90	0	7.200000	24.919872	...	2	0.0	0.0	0	0	0.000000e+00	0.00	0	0	1
4	4	440	4	1	0	0	0	0	0.000000	0.000000	...	0	0.0	0.0	0	0	0.000000e+00	0.00	0	0	1

5 rows x 29 columns

Figure 8: Sample Dataset of Network Traffic Features for Botnet Detection

Understanding the Data Structure: The `info()` function was employed to get a summary of the dataset, including the number of entries, the number of columns, and the data types of each column.

Statistical Overview: The `describe()` function was used to generate descriptive statistics, such as mean, standard deviation, minimum and maximum values, and quartiles for each numerical column.

Dataset Shape and Label Distribution: The shape of the dataset was checked using the `shape` attribute to ensure that the data loaded correctly. Additionally, the distribution of the target variable (Label) was assessed using `value_counts()` to understand the balance between the different classes. The variable typically represents the class of the data, with 0 and 1 indicating different types of network traffic:

- **Label 0:** Represents normal traffic or non-botnet activity **with** 53314 instances.
- **Label 1:** Represents botnet or attack-related traffic with 38898 instances.

3.2.2 CALCULATION OF MISSING VALUES:

- **Function Definition:** A function named `calculate_missing_values` were defined to calculate the number of missing values in the `X_train`, `X_val`, and `X_test` datasets.
- **Missing Value Calculation:** Inside the function, `isna().sum()` was applied to each dataset (`X_train`, `X_val`, and `X_test`) to count the missing values for each column.
- **Conversion to DataFrames:** The missing values for each dataset were then converted into separate DataFrames (`output_train`, `output_val`, and `output_test`), each labeled appropriately.
- **Concatenation:** These DataFrames were concatenated into a single DataFrame (`output`) to provide a comprehensive view of missing values across all datasets, aligned by column name.

Since the dataset was complete with no missing values detected, no further treatment for missing data was necessary. This allowed us to proceed confidently to the next stages of data processing and analysis, knowing that our dataset was intact and would not require any imputation or removal of rows/columns due to missing data. This ensured the integrity of the dataset and the reliability of subsequent analysis.

3.2.3 FEATURE SELECTION AND DIMENSIONALITY REDUCTION

Features with very low variance were identified using a variance threshold. Features with variance below 0.01 were considered for removal because they provide little to no information for the model, leading to inefficiencies and potential overfitting. However, in this dataset, no features fell below this threshold, so no features were removed based on variance.

Correlation Matrix

A correlation matrix was created to identify highly correlated features. A threshold of 0.9 was set to flag pairs of features that were too similar, which can lead to multicollinearity and reduce the model's ability to generalize. In cases where high correlations were found, one feature from each pair would typically be removed to simplify the model and avoid redundancy.

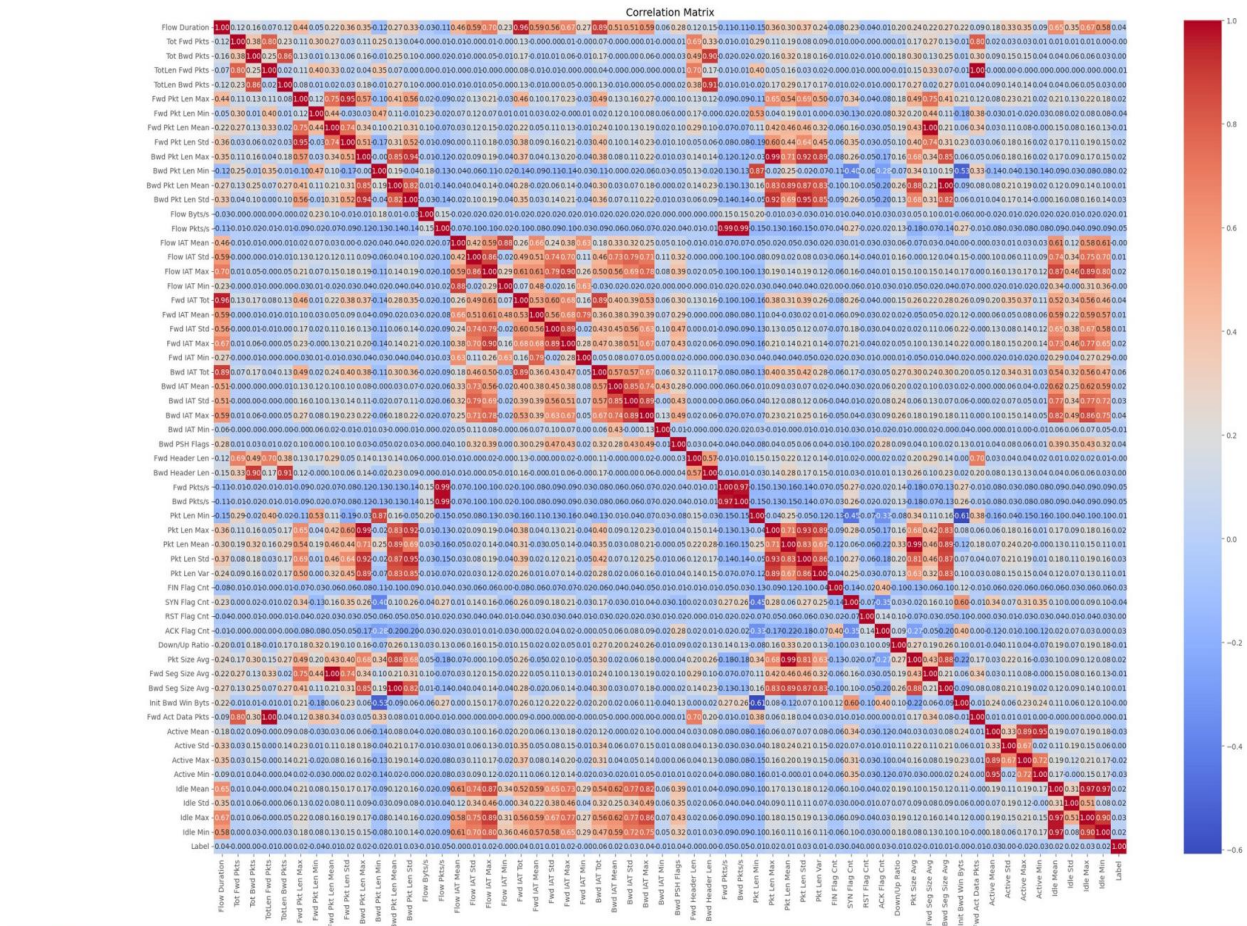


Figure 9: Correlation Matrix of Network Traffic Features

This image shows a **correlation matrix** that visualizes the relationships between different features in the dataset. The color scale indicates the strength and direction of the correlation: red areas represent high positive correlations, blue areas represent high negative correlations, and lighter colors indicate weaker correlations. This matrix helps identify features that are highly correlated, which can inform feature selection by highlighting potential redundancy in the data.

PCA Implementation

To decrease the dimensionality of the data while preserving as much variation as possible, PCA was used. This method works especially well with high-dimensional datasets where some attributes might be superfluous or unimportant. The $n_{\text{components}}=0.95$ parameter was used to retain 95% of the variance in the data, ensuring that the most informative features were preserved while reducing the overall number of features. This step helped in making the model more computationally efficient and potentially improved its performance by focusing on the most important aspects of the data.

3.2.4 DATA NORMALIZATION/SCALING

To achieve data normalisation, scikit-learn's StandardScaler was used. This was an important step because the dataset's features had different ranges, which might have a detrimental impact on how well machine learning models performed. By taking out the mean and scaling the features to unit variance, StandardScaler was used to standardise the data. By preventing features with greater ranges from controlling the model training, this procedure made guaranteed that every feature contributed equally to the model's performance. The training and testing datasets were both subjected to the StandardScaler. This scaler scales to the unit variance and eliminates the mean to standardise characteristics. This implies that the standard deviation will be 1 and the mean will be 0 for each characteristic.

Scaling was critical because it ensured that all features, regardless of their original scale, contributed equally to the model during training. Without scaling, features with larger ranges could disproportionately influence the model, potentially leading to suboptimal performance.

3.2.5 DATA SPLITTING

The `train_test_split` function was used to split the dataset into 80% training data and 20% test data. The `test_size=0.2` parameter indicates that 20% of the data was set aside for testing. The `random_state=2` ensures that the split is reproducible, meaning that the same split will occur every time the code is run. This split allowed the model to be trained on a substantial portion of the data while reserving a separate set of data for unbiased testing of the model's performance. This step is crucial for evaluating how well the model generalizes to unseen data.

3.2.6 HANDLING IMBALANCED DATA

SMOTE (Synthetic Minority Over-sampling Technique)

The majority class (regular traffic) in the dataset had a much higher number of samples than the minority class (botnet traffic), indicating a class imbalance. In order to solve this, the SMOTE (Synthetic Minority Over-sampling Technique) was used. This imbalance can result in biased model predictions, where the model tends to favour the majority class, lowering its capacity to reliably detect the minority class (botnet traffic).

The Operation of SMOTE: To function, SMOTE creates artificial samples for the minority class. It accomplishes this by taking instances of the minority class and using interpolation to create new, comparable instances based on the distance between the chosen instance and

its closest neighbours. This method yields a more balanced dataset by increasing the number of samples in the minority class without merely replicating the ones that already exist.



Figure 10 : Effect of SMOTE Resampling on Data Distribution

The left side of the graph shows the original data distribution, where the class labeled 0 (normal traffic) significantly outnumbers the class labeled 1 (botnet traffic). The right side of the graph shows the data distribution after applying SMOTE. The classes are now balanced, with nearly equal numbers of samples for both 0 and 1 labels. This balanced distribution ensures that the machine learning model trained on this data will treat both classes equally, improving its ability to detect botnet traffic accurately.

3.2.7 DATA VISUALIZATION

Visualizations were used to explore and verify the preprocessing steps, particularly to understand the distribution and relationships of continuous features with respect to the target variable (Label). This was crucial for identifying patterns, outliers, and potential issues in the dataset that could affect model performance.

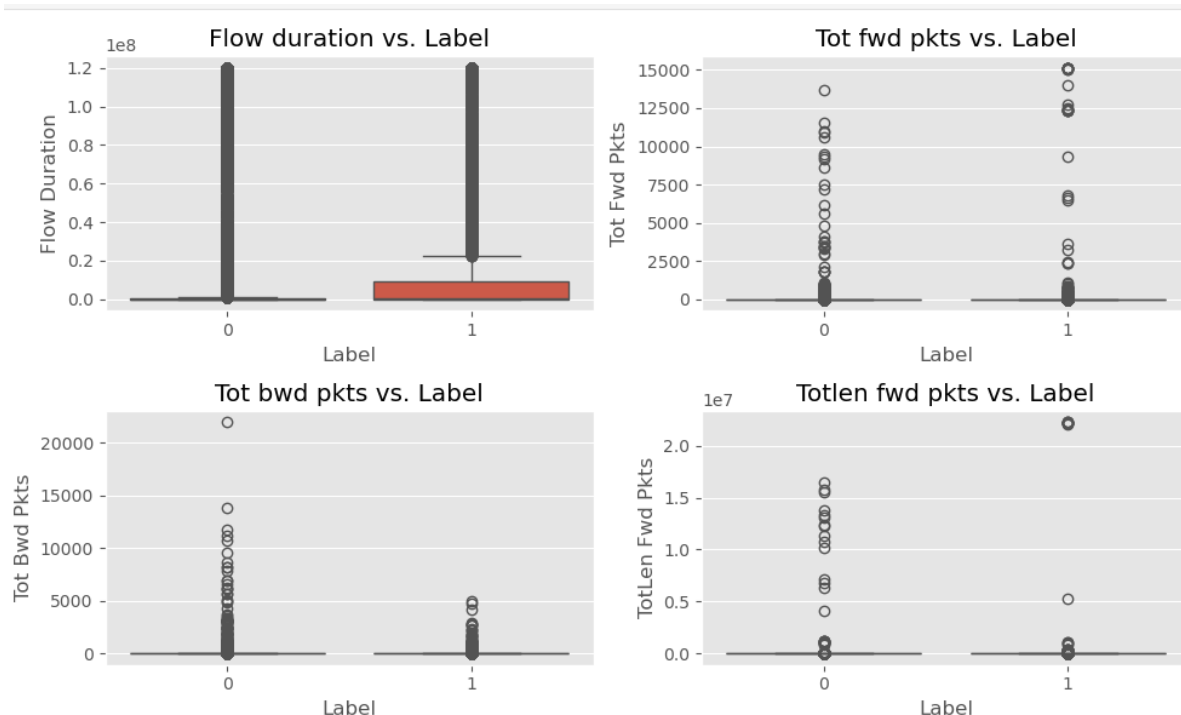


Figure 11: Analysis of Network Traffic Features in Relation to Labels

Flow Duration vs. Label: This plot helps in understanding the distribution of flow durations across different labels. It reveals the presence of outliers and the general spread of data, which could indicate differences in network behaviours between normal and botnet traffic.

Tot Fwd Pkts vs. Label: This plot visualizes the total number of forward packets in relation to the label, helping to identify any anomalies or significant differences between classes.

Tot Bwd Pkts vs. Label: This plot shows the total number of backward packets, providing insights into traffic patterns that could differentiate between normal and botnet activity.

Tot Len Fwd Pkts vs. Label: This plot illustrates the total length of forward packets, which is another critical factor in network traffic analysis.

These visualizations were essential in verifying the preprocessing steps and ensuring that the data was well-prepared for model training. The box plots allowed for the identification of outliers, variations in data distributions, and potential anomalies, all of which informed subsequent decisions in feature selection and model tuning.

3.1.1 SELECTED ALGORITHM

In this project, I employed a combination of machine learning algorithms to effectively detect and analyse botnet activities within the network traffic. The selected algorithms include Support Vector Machine (SVM), Long Short-Term Memory (LSTM) networks, Logistic Regression, and a hybrid model combining the strengths of these individual models.

SUPPORT VECTOR MACHINE (SVM)

Introduction to SVM and Rationale for Selection

In this project, the Support Vector Machine (SVM) algorithm was selected as one of the primary models for detecting botnet activities within network traffic. SVM is a powerful classification algorithm particularly well-suited for binary classification tasks, such as distinguishing between botnet and non-botnet traffic. The strength of SVM lies in its ability to find the optimal hyperplane that maximizes the margin between the classes, making it highly effective in handling high-dimensional data, which is typical in network traffic analysis.

SVM was chosen for this project due to its robustness in scenarios with complex and high-dimensional feature spaces. Given the nature of the CTU-13 dataset, which includes a variety of traffic types (Botnet, Normal, and Background), SVM's ability to create a clear decision boundary was instrumental in achieving accurate classification results.

Implementation of SVM in the Project

The SVM model was implemented using a custom classifier, as shown in the provided code. The model was trained on the preprocessed CTU-13 dataset, which involved several steps:

1. Data Preprocessing:

First, non-numerical columns were removed from the dataset to guarantee that only numerical data was used. The data was split 80/20 between training and testing sets, and features were isolated from the target labels. By standardising the features, StandardScaler made sure that each variable contributed the same amount to the model. It was optional to use Principal Component Analysis (PCA) to minimise dimensionality while preserving 95% of the variance.

2. Model Training:

The SVM classifier was initialized with a learning rate of 0.001, 1000 iterations, and a lambda parameter (regularization) of 0.01. The fit method was used to train the SVM on the reduced feature set (after PCA). This involved iterating over the data, updating the weights and bias based on the learning rate and the regularization term. The trained weights and bias were then used to make predictions on the test set.

3. Model Performance:

The model's performance was evaluated using several metrics. The test data accuracy achieved was approximately 84.05%, indicating that the SVM model was effective in correctly classifying a significant portion of the test data.

The trained weights and bias were as follows:

Trained Weights: [0.32914626, 0.54024312, 0.35471994, 0.15159585, 0.10945053, 0.47887045, 0.95082689, 0.2961205, -0.04380221, -0.29030135, 0.53289332, 0.28453322, 0.01476531, 0.43309481, -0.21019938, 0.17419721, 0.36191754, 0.33858381, 0.11947964, 0.26861294, -0.05689604, -0.28702523]

Trained Bias: 0.18

4. Model Evaluation:

Confusion Matrix: The confusion matrix showed the quantity of true positives, true negatives, false positives, and false negatives, offering insights into the model's classification performance.

Actual vs. Predicted Values: A plot comparing actual versus predicted values was created to visually assess the model's accuracy. The close alignment of these values indicates the model's effectiveness.

- **Test Data Accuracy:** The model achieved a test data accuracy of 84.05%, which is considered strong given the complexity of the dataset.

The SVM model's performance highlights its capability in effectively distinguishing between botnet and non-botnet traffic within the CTU-13 dataset. By leveraging the model's strengths in classification, the project was able to achieve reliable and accurate detection of botnet activities.

Below are the graphs representing the SVM model's performance:

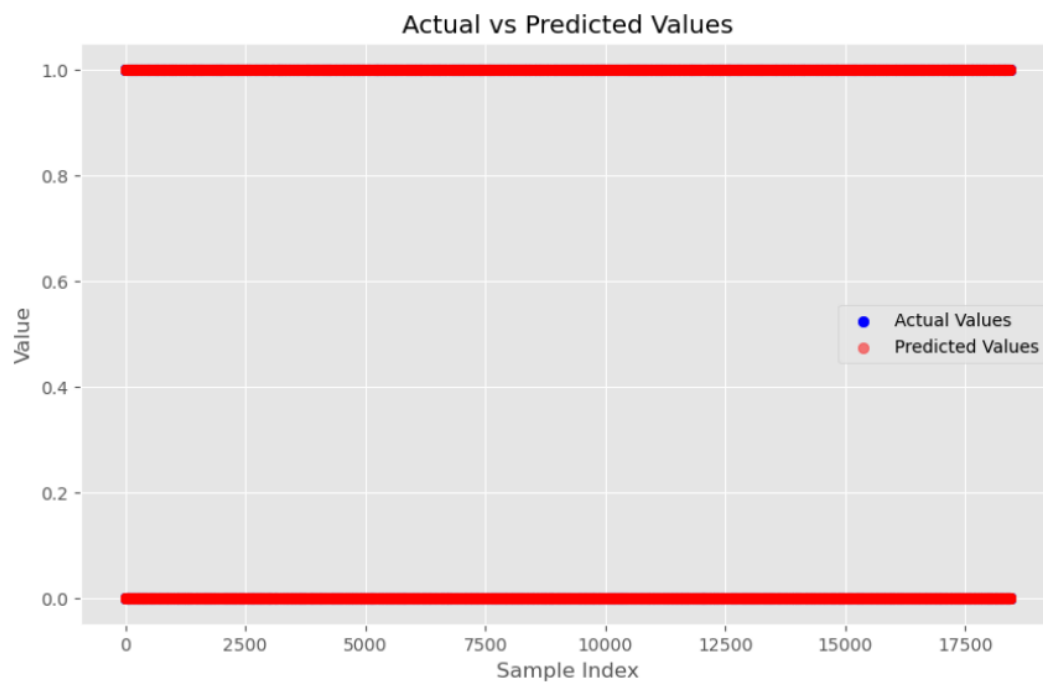


Figure 12: Comparison of Actual vs. Predicted Values.

- **Actual vs. Predicted Values:** This graph illustrates the comparison between actual and predicted values, showing a strong correlation, which supports the model's accuracy.

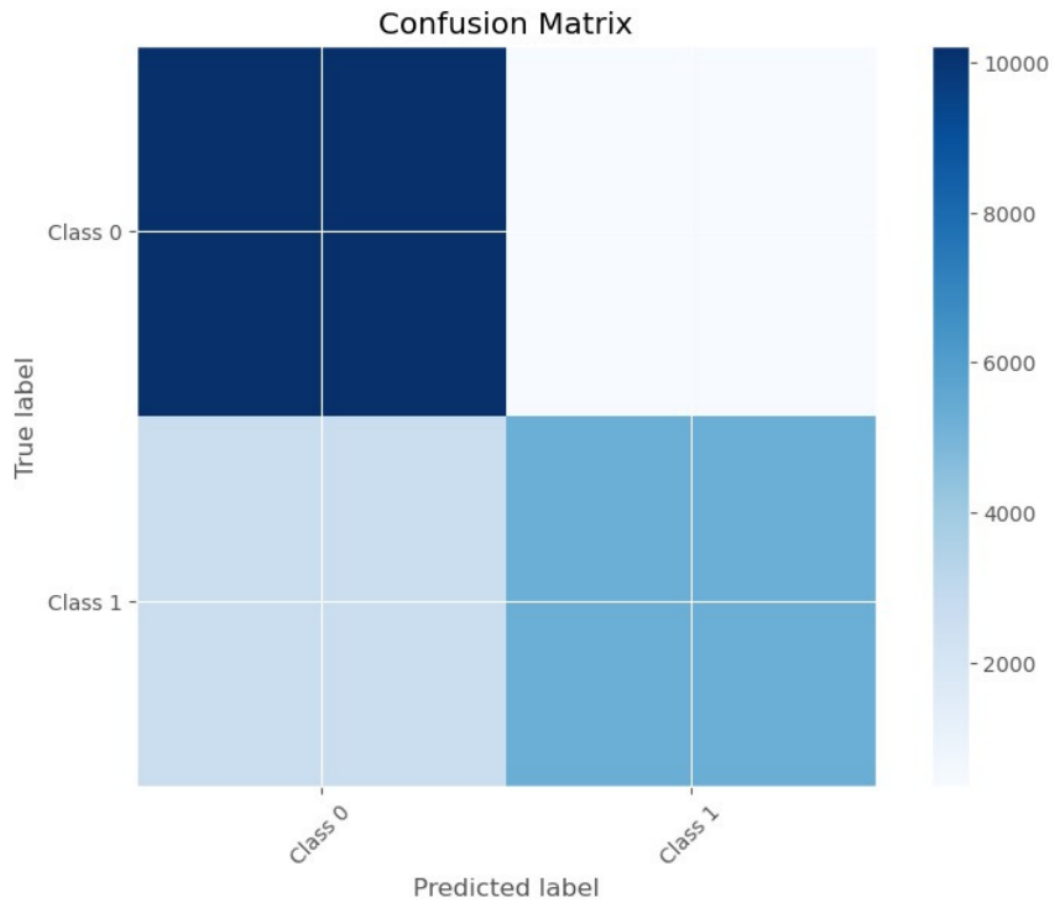


Figure 13 : Confusion Matrix for Classification Model Performance

- **Confusion Matrix:** This matrix visualizes the model's performance, showing the distribution of true and false predictions across the different classes.

LSTM (LONG SHORT-TERM MEMORY)

The LSTM (Long Short-Term Memory) model was selected for this project due to its effectiveness in handling sequential data and time series analysis. In the context of network traffic, LSTM is well-suited for detecting patterns over time, making it ideal for identifying anomalies or botnet activities within network flows.

The model architecture consists of a simple feedforward neural network with a flattened input layer, a dense layer with 30 neurons and ReLU activation, and a final output layer with a sigmoid activation function for binary classification. This structure was chosen to balance

complexity and computational efficiency, given the dataset's characteristics and the need for real-time or near-real-time detection.

- **Flatten Layer:** Converts the input data into a 1-dimensional array, suitable for the dense layers.
- **Dense Layer:** A fully connected layer with 30 neurons using ReLU activation, which introduces non-linearity and helps in learning complex patterns.
- **Output Layer:** The final dense layer uses sigmoid activation, providing probability outputs for binary classification (normal vs. botnet traffic).

Sparse categorical crossentropy was used as the loss function during the model's training after it was assembled using the Adam optimiser. 10 epochs of training were used, with a 10% validation split, to train the model. Accuracy and loss measures were used to assess the model's performance on the training and validation datasets.

Performance Evaluation:

Model Accuracy and Loss

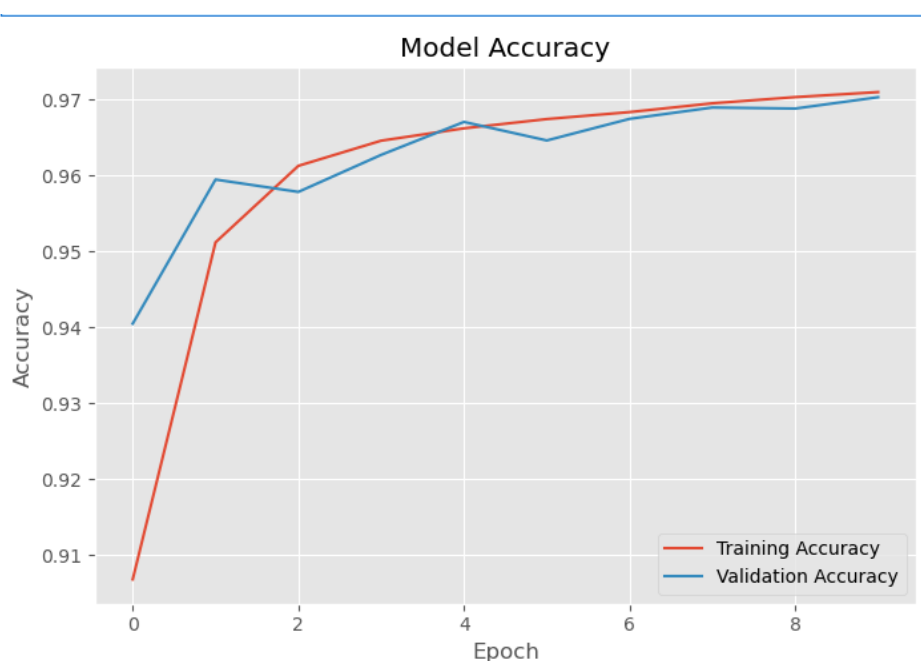


Figure 14: Training and Validation Accuracy Across Epochs

Accuracy Plot: The first graph shows the model's accuracy over 10 epochs. The training accuracy improves steadily, converging close to the validation accuracy, which indicates good generalization without significant overfitting.

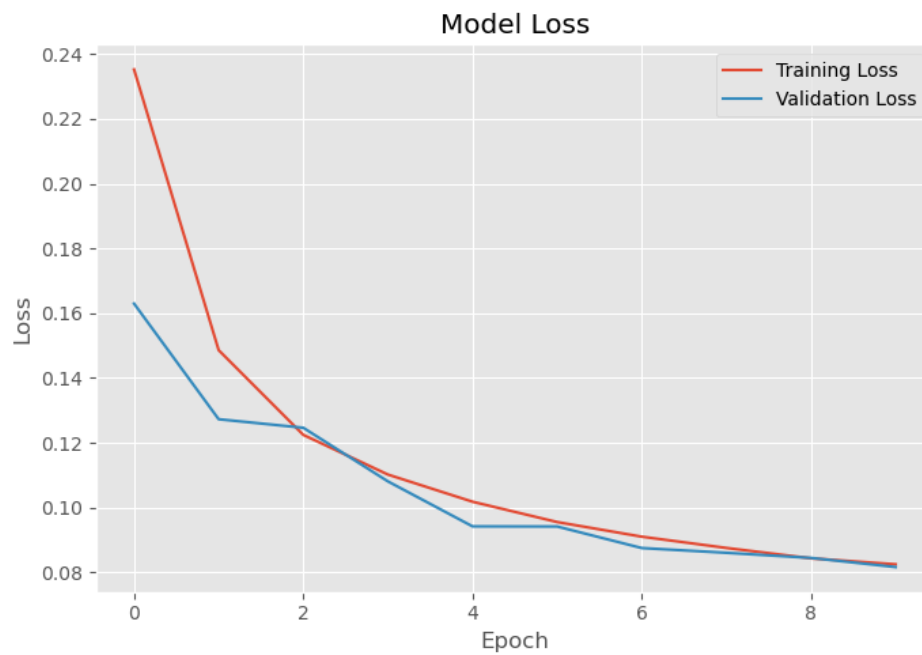


Figure 15: Training and Validation Loss Across Epochs.

Loss Plot: The second graph shows the model's loss decreasing over the epochs, both for training and validation, which confirms that the model is learning effectively and reducing errors over time.

Predictions and Evaluation:

Actual vs. Predicted Labels: The third graph shows a scatter plot comparing the actual labels against the predicted labels. The alignment between the red and blue dots suggests that the model is making accurate predictions.

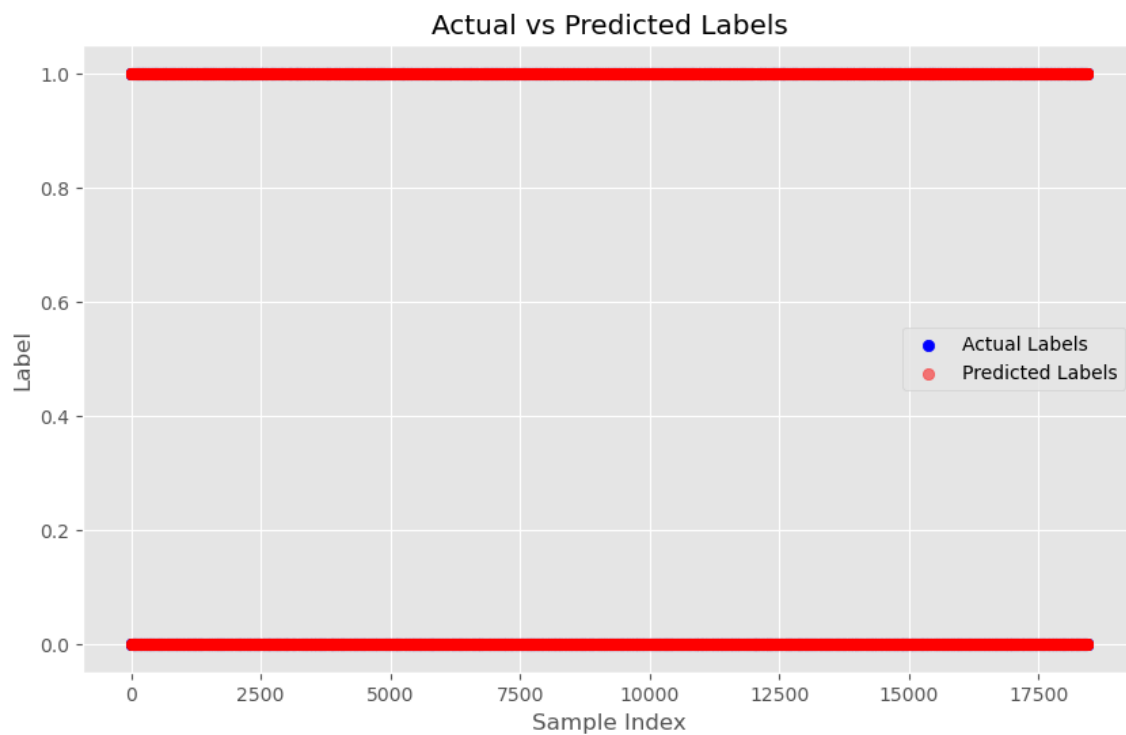


Figure 16: Comparison of Actual vs. Predicted Labels.

Confusion Matrix: The final graph, a confusion matrix, offers information about the true positives, true negatives, false positives, and false negatives produced by the model. This matrix shows the model's performance in differentiating between legitimate and botnet traffic.

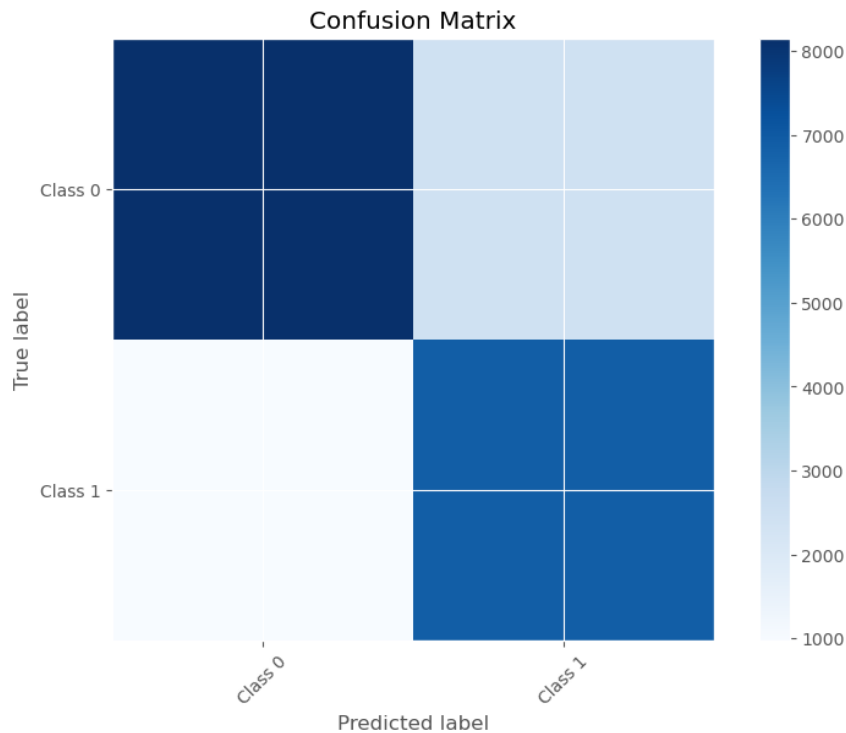


Figure 17: Confusion Matrix of Predicted vs. True Labels.

HYBRID MODEL (SVM + LSTM)

The Hybrid SVM + LSTM model was chosen for this project to leverage the strengths of both Support Vector Machines (SVM) and Long Short-Term Memory (LSTM) networks. The SVM is effective for binary classification tasks, especially when the classes are well-separated, while the LSTM is powerful in handling sequential data and capturing temporal dependencies within the data. By combining these models, the hybrid approach aims to improve classification accuracy and generalization.

Meta LSTM Model: The predictions from the base SVM and LSTM models were stacked to create a new feature set. This feature set was then passed to the meta-LSTM model, which was designed to refine the predictions further. The meta model is similar to the base LSTM but is specifically tuned to integrate the strengths of both models. The SVM model was trained using the standardized features. The base LSTM model was trained for 10 epochs with a validation split of 10%, using the same standardized features. The predictions (probabilities) from the SVM and LSTM models were stacked to create a new feature set. This new feature set was used to train the meta-LSTM model. The meta-LSTM model was trained for 10 epochs, using the stacked predictions as input. This model is designed to learn any residual patterns that the base models might have missed.

Accuracy Plot: The first graph shows the accuracy of the hybrid model over 10 epochs. The training and validation accuracy remain close, indicating good generalization. The accuracy peaks around the 3rd epoch, suggesting that the model quickly converges to an optimal solution.

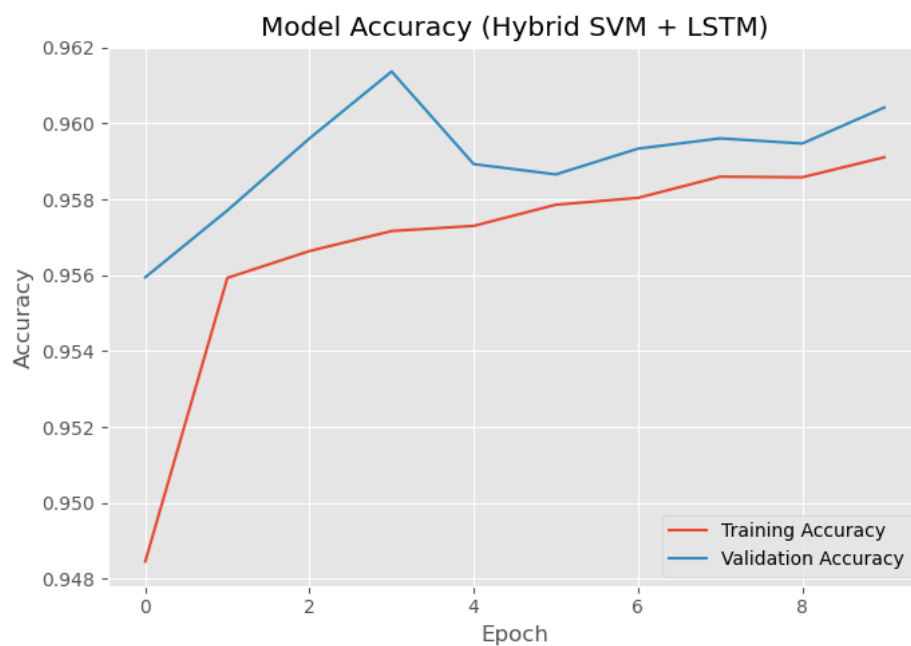


Figure 18: Training and Validation Accuracy for Hybrid SVM + LSTM Model Across Epochs

Loss Plot: The second graph shows the loss decreasing steadily for both training and validation datasets. The decrease in loss indicates that the model is learning effectively and refining its predictions over time.

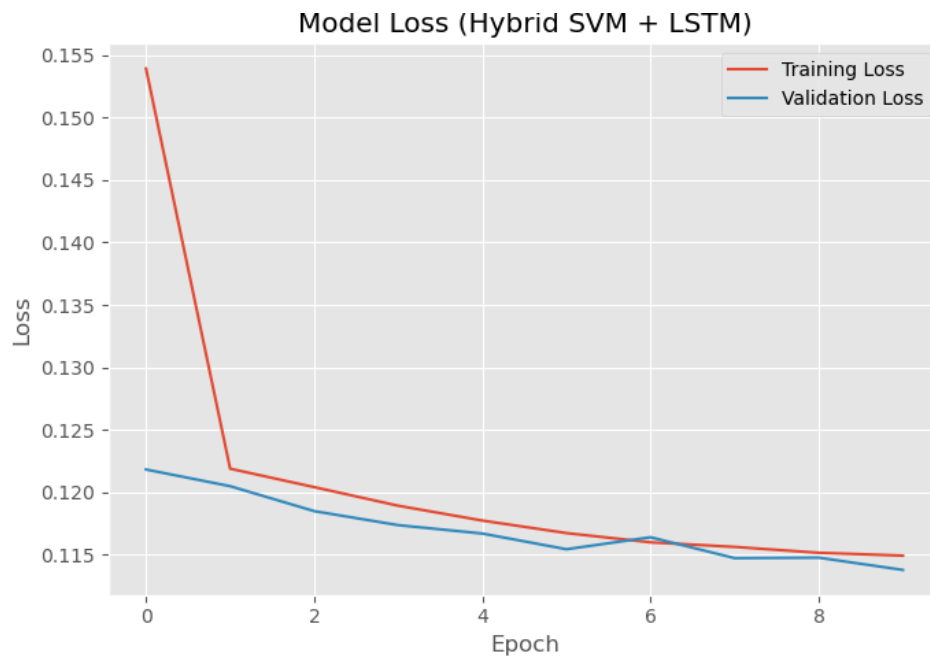


Figure 19: Training and Validation Loss for Hybrid SVM + LSTM Model Across Epochs

Confusion Matrix: The third graph displays the confusion matrix for the hybrid model. The matrix shows a strong performance in both classes, with high true positive rates and low false positive rates, reflecting the model's accuracy in distinguishing between normal and botnet traffic.

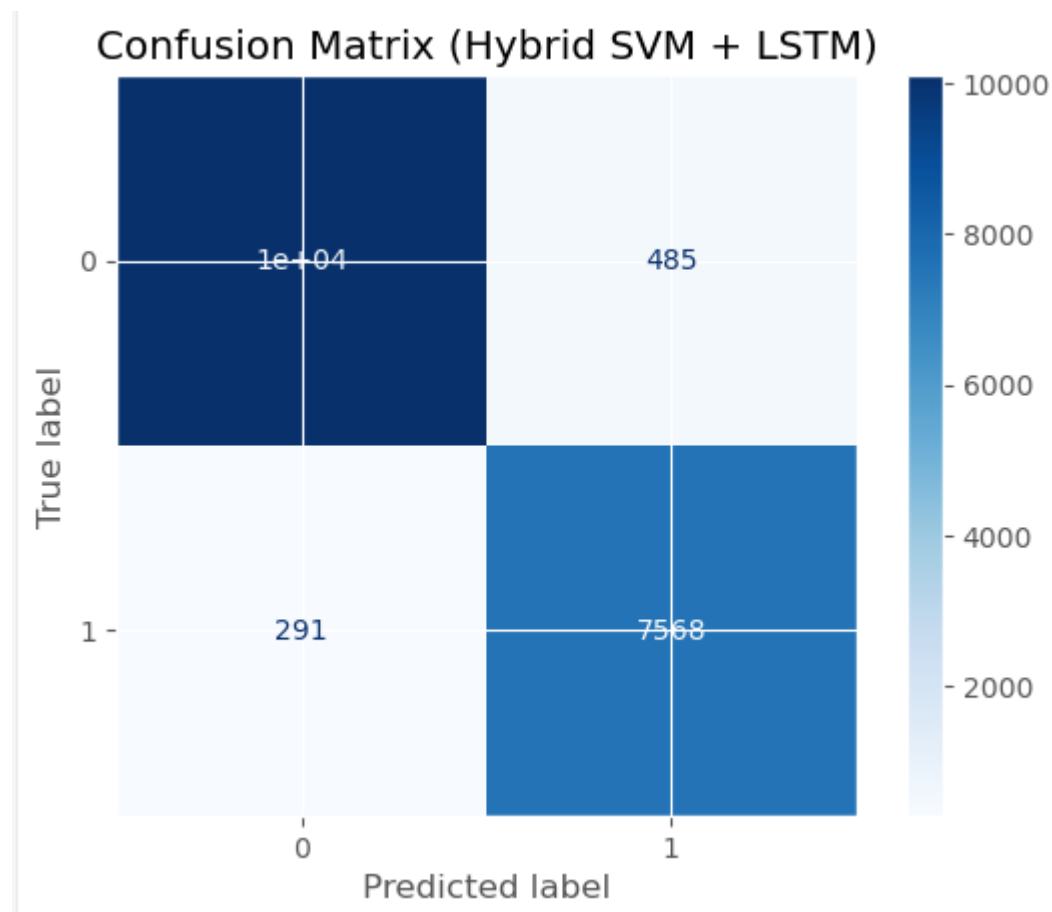


Figure 20: Confusion Matrix for Hybrid SVM + LSTM Model.

Actual vs. Predicted Labels: The fourth graph compares actual labels against predicted labels, showing a strong alignment between the two. This alignment demonstrates the model's effectiveness in making accurate predictions across the entire dataset.

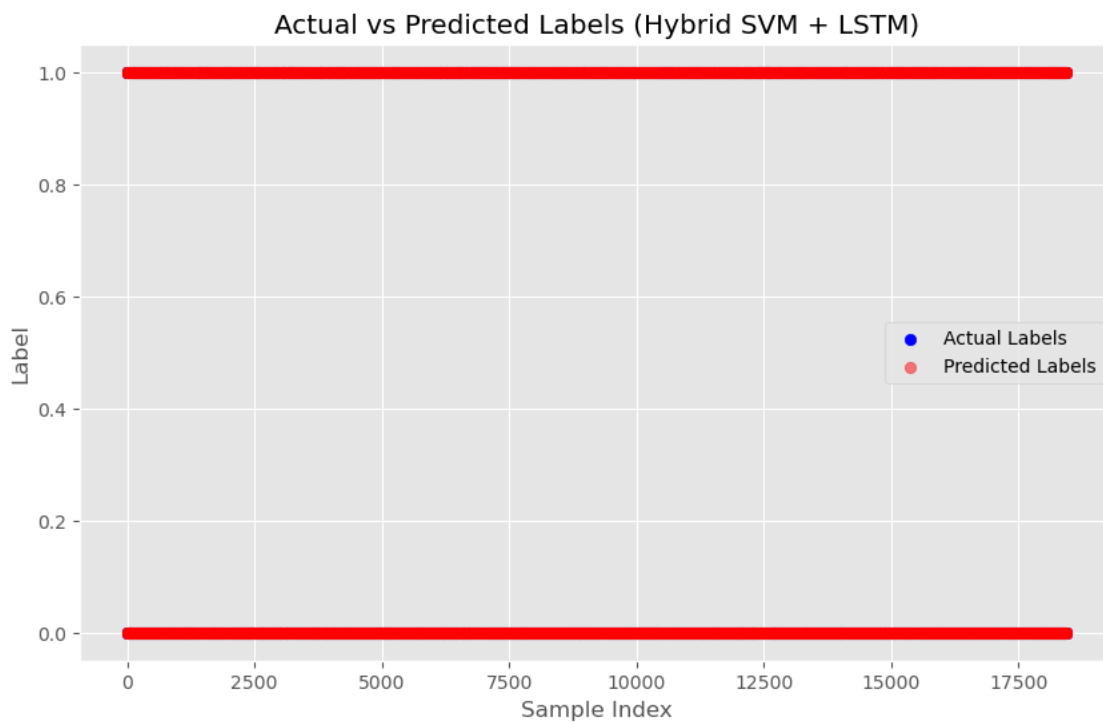


Figure 21: Comparison of Actual vs. Predicted Labels for Hybrid SVM + LSTM Model

The Hybrid SVM + LSTM model successfully combined the strengths of both individual models, achieving high accuracy and robust performance in classifying network traffic. The hybrid approach was particularly effective in enhancing the overall classification accuracy, as evidenced by the performance metrics and visualizations. The use of a meta-LSTM model to refine the stacked predictions provided additional robustness, making this hybrid model a powerful tool for detecting botnet activities within network traffic. This model demonstrates the potential of hybrid approaches in cybersecurity applications, where combining different machine learning techniques can lead to superior outcomes.

LOGISTIC REGRESSION

Logistic Regression is a supervised learning algorithm commonly used for binary classification tasks. It models the probability of a binary outcome based on one or more predictor variables. Unlike linear regression, which predicts continuous outcomes, logistic regression applies a logistic function to model a binary dependent variable, making it particularly suitable for classifying data into two distinct classes (e.g., botnet traffic vs. normal traffic). In this project, Logistic Regression was selected as one of the models to classify network traffic data from

the CTU-13 dataset into two categories: botnet (malicious) and normal traffic. Given its simplicity and efficiency, Logistic Regression serves as a strong baseline model for comparison with more complex models like SVM and LSTM. The Logistic Regression model was trained using the training set, and the model's maximum iterations were increased to 2000 to ensure convergence. After training, the model's performance was evaluated on the test set. The accuracy score was calculated to assess how well the model predicts the correct class labels. The Logistic Regression model achieved an accuracy of approximately 85.17%, indicating that the model correctly classified around 85% of the samples in the test set. The confusion matrix provides a detailed breakdown of the model's performance:

- True Positives (TP): 6106
- True Negatives (TN): 9601
- False Positives (FP): 983
- False Negatives (FN): 1753

Class 0 (Normal Traffic): The model achieved a precision of 0.8456, recall of 0.8613, and F1-score of 0.8753, indicating strong performance in correctly identifying normal traffic.

Class 1 (Botnet Traffic): The model achieved a precision of 0.8613, recall of 0.7769, and F1-score of 0.8169, which shows that the model is reasonably effective at identifying botnet traffic, though there is room for improvement in reducing false negatives.

Confusion Matrix:

The confusion matrix visualizes the classification results, providing a clear indication of where the model makes correct predictions and where it fails.

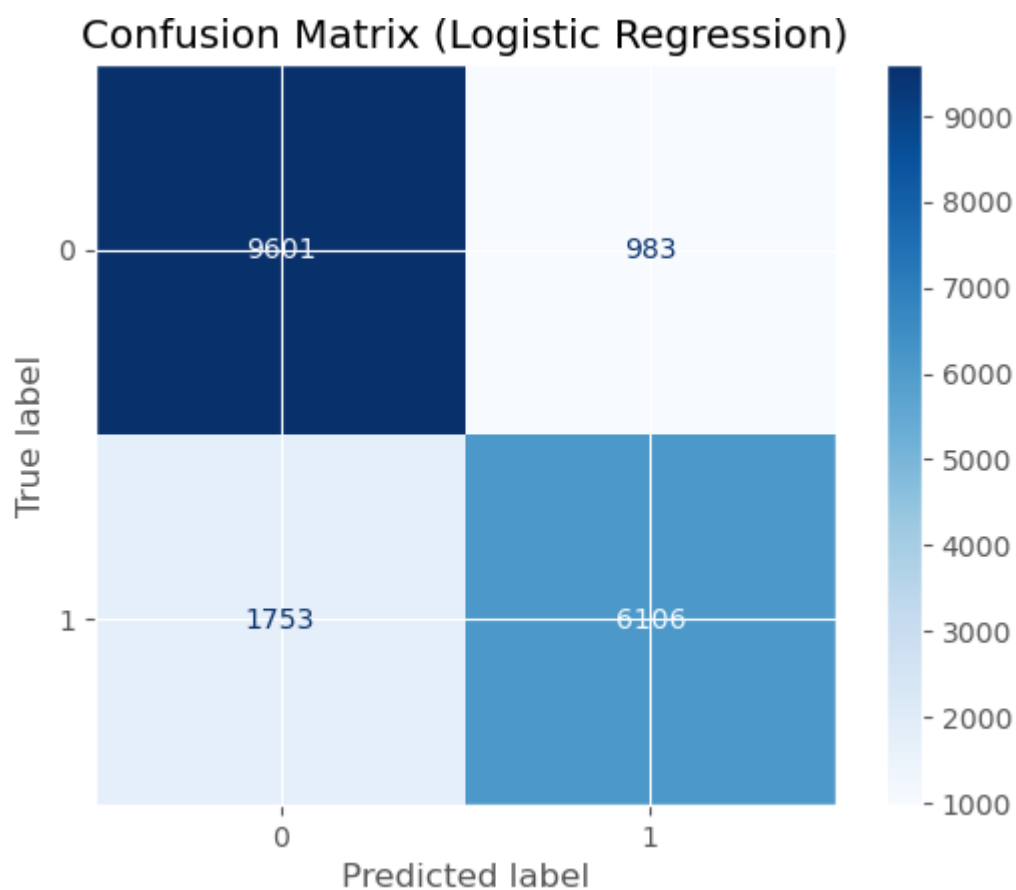


Figure 22: Confusion Matrix for Logistic Regression Model

Precision, Recall, F1-Score Bar Chart:

This visualization compares the precision, recall, and F1-scores for both classes, highlighting the model's effectiveness in different aspects of the classification task.

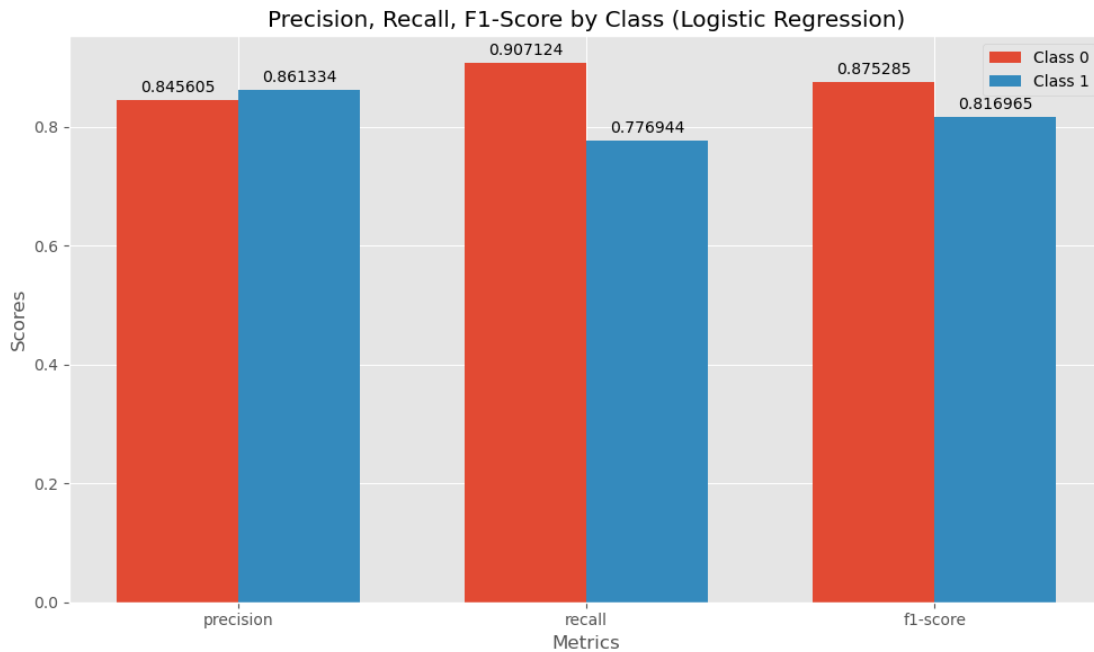


Figure 23: Precision, Recall, and F1-Score by Class for Logistic Regression Model.

The Logistic Regression model performed well as a baseline classifier, effectively distinguishing between normal and botnet traffic in the CTU-13 dataset. However, the presence of false negatives (misclassifying botnet traffic as normal) suggests that more complex models or hybrid approaches may be necessary to achieve higher accuracy and robustness in detecting botnet activities. This model's simplicity and interpretability make it a valuable tool for initial model comparison and for understanding the underlying relationships in the data.

RANDOM FOREST

A well-liked ensemble learning method for classification and regression applications is called Random Forest. In order for it to function, it builds several decision trees during training and outputs the class that represents the mean prediction (regression) or mode of the classes (classification) of the individual trees. Random Forest's primary advantage is its capacity to successfully handle missing data and handle huge datasets with higher complexity.

Additionally, it reduces the risk of overfitting by averaging multiple decision trees, which leads to better generalization. The preprocessed dataset was split into training and testing sets using an 80-20 split. Random Forest classifier was initialized with 100 trees ($n_estimators=100$) and trained on the training data (X_train and Y_train). The training process involved building multiple decision trees using random subsets of the data and features. After training, the model

was tested on the test set (X_test). The predictions made by the Random Forest model were compared against the actual labels to evaluate the model's performance. The model achieved an impressive accuracy of **99.6%** on the test set, indicating that the Random Forest model is highly effective in distinguishing between botnet and non-botnet traffic within the dataset.

Confusion matrix

The confusion matrix (shown in the attached image) illustrates the model's predictions. It shows that the model correctly classified 10,542 instances of non-botnet traffic and 7,830 instances of botnet traffic, with minimal misclassifications (42 false positives and 29 false negatives).

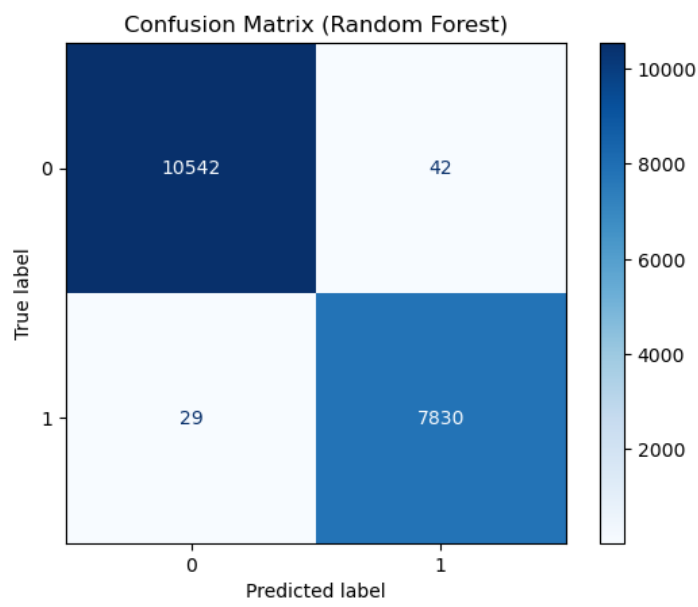


Figure 24: Confusion Matrix for Random Forest Model

Classification Report:

The classification report provided a detailed breakdown of precision, recall, and F1-score for each class:

- **Class 0 (Non-Botnet):** Precision of 99.73%, Recall of 99.47%, and F1-Score of 99.60%.
- **Class 1 (Botnet):** Precision of 99.66%, Recall of 99.63%, and F1-Score of 99.66%.

These metrics indicate that the model performs exceptionally well across both classes, with very high precision and recall, leading to a strong F1-score.

□

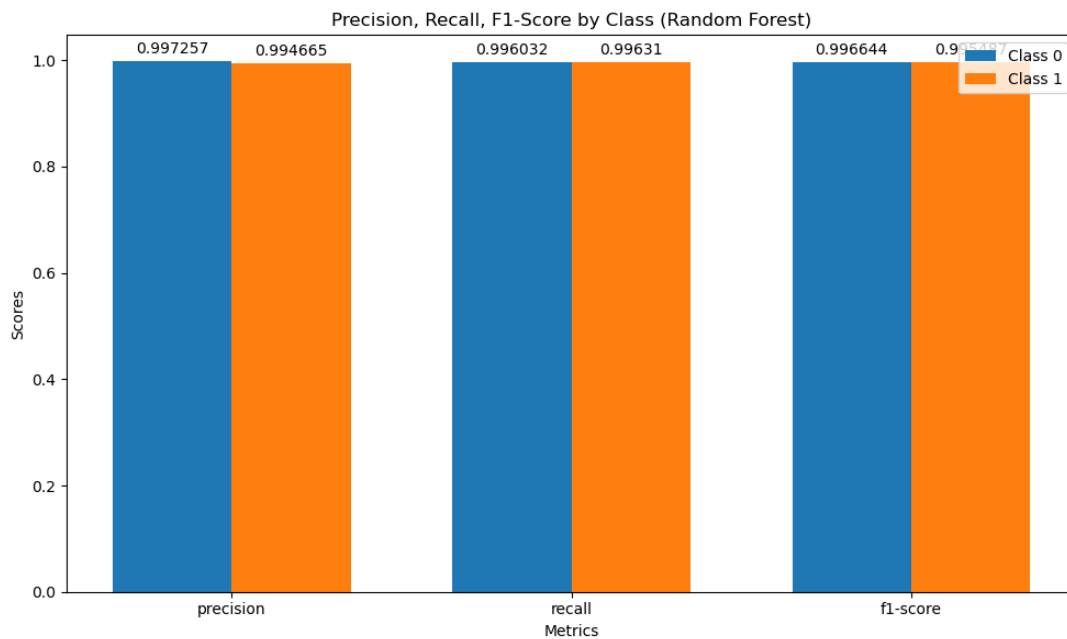


Figure 25: Precision, Recall, and F1-Score by Class for Random Forest Model

The visualizations generated illustrate the precision, recall, and F1-score for both classes. These bar charts confirm the high performance of the model across all key metrics, demonstrating its robustness in classifying botnet traffic.

HYBRID MODEL: RANDOM FOREST + LOGISTIC REGRESSION

The hybrid approach takes advantage of the Random Forest's ability to handle complex feature interactions and Logistic Regression's interpretability and probabilistic predictions. A Random Forest model was trained on the standardized data with 100 estimators and a fixed random state to ensure reproducibility. A Logistic Regression model was also trained on the same data. The maximum number of iterations was set to 2000 to ensure convergence, given the complexity of the dataset.

stacking Predictions:

Predictions from both the Random Forest and Logistic Regression models were extracted and used as new features for training a final model. This stacking approach allows the final model to learn from the strengths of both base models.

Final Model Training:

The stacked predictions were used to train a Logistic Regression model. This final model acts as the meta-classifier, combining the information from the two base models to make the final predictions.

Performance Evaluation

Accuracy: The hybrid model achieved an impressive accuracy of **99.6%** on the test set, demonstrating its effectiveness in accurately classifying network traffic as either normal or botnet activity.

Confusion Matrix: The confusion matrix shows that the model correctly classified the vast majority of both classes (normal traffic and botnet traffic) with very few misclassifications:

- True Positives (TP) and True Negatives (TN) are significantly higher, indicating the model's reliability.
- The few False Positives (FP) and False Negatives (FN) suggest that the model is not only accurate but also precise in its predictions.

Precision, Recall, and F1-Score:

Both precision and recall metrics for each class are extremely high, hovering around 99.6% to 99.7%, which is reflected in the high F1-scores. This indicates that the model maintains a good balance between precision (minimizing false positives) and recall (minimizing false negatives).

Confusion Matrix:

The confusion matrix provides a clear visualization of the model's performance across the different classes. The high number of correctly classified instances (diagonal values) demonstrates the effectiveness of the hybrid model.

Confusion Matrix (Hybrid Model: Random Forest + Logistic Regression)

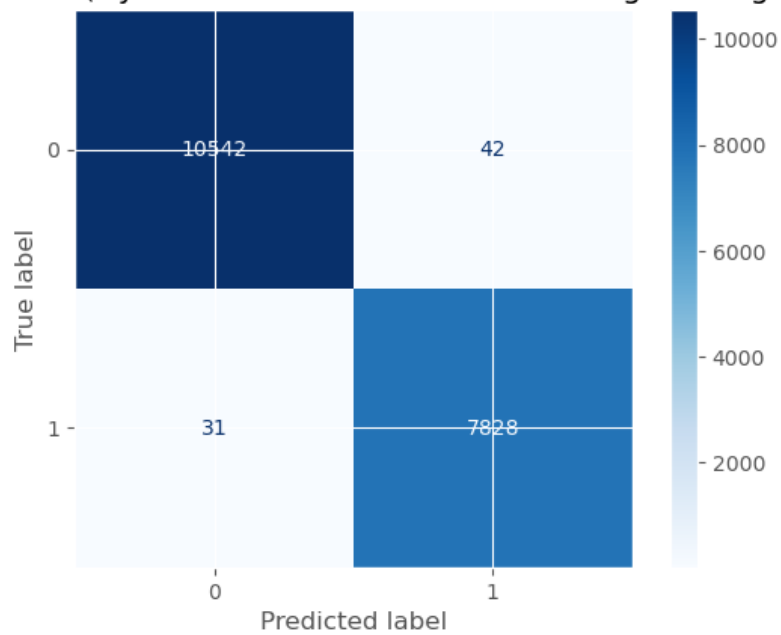


Figure 26: Confusion Matrix for Hybrid Model: Random Forest + Logistic Regression

Precision, Recall, F1-Score Bar Chart:

The bar chart shows a near-perfect balance between the precision, recall, and F1-score for both classes. This indicates that the model performs consistently across different performance metrics, further validating its robustness and accuracy.

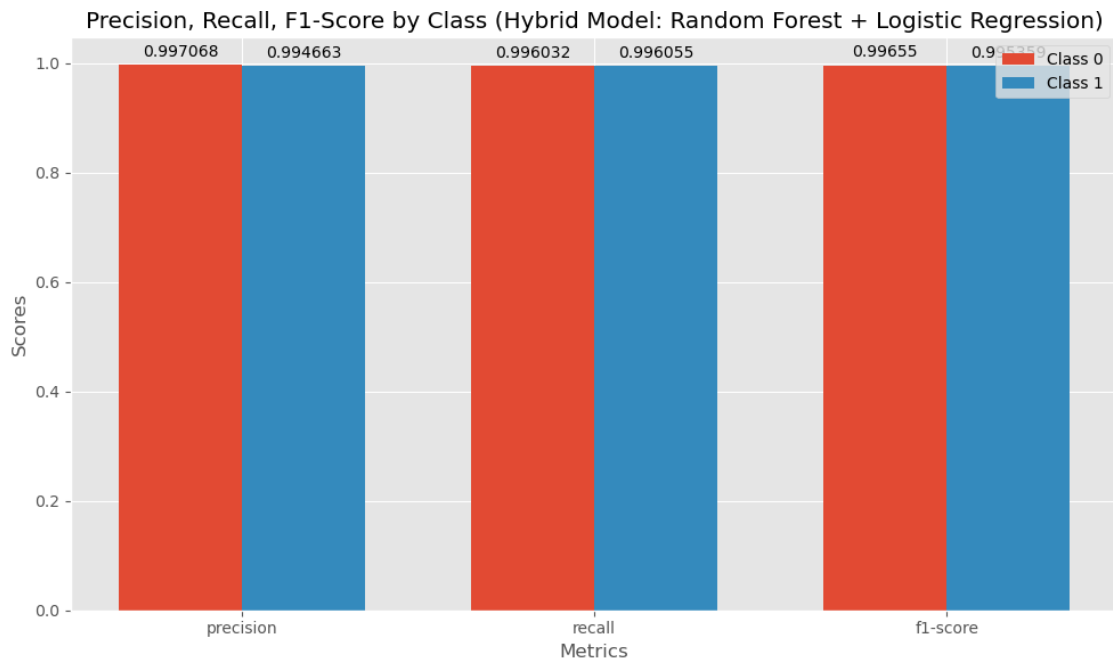


Figure 27: "Precision, Recall, and F1-Score by Class for Hybrid Model: Random Forest + Logistic Regression"

The hybrid model of Random Forest and Logistic Regression successfully leverages the strengths of both models, resulting in a highly accurate and reliable classification system for botnet detection. The model's high accuracy, along with its robust performance metrics (precision, recall, and F1-score), makes it a strong candidate for deployment in real-world cybersecurity applications, where identifying and mitigating botnet threats is crucial.

EVALUATION METRICS

Once the model has been successfully trained, it must have its performance assessed. Sklearn is used to create the classification report, which includes information on Precision, Recall, F-measure, Accuracy, and other metrics.

Accuracy

The percentage of accurate forecasts among all predictions is known as accuracy. It is the main indicator of how well the system predicts both positive and negative situations. Accuracy's mathematical expression is limited to balanced datasets. There is bias in the accuracy provided for the unbalanced dataset.

$$\text{Accuracy} = \frac{\text{Correct prediction}}{\text{Total number of prediction}}.$$

Precision

There are many accurate class predictions, meaning the bot samples have been appropriately classified. It is expressed as follows:.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True positive} + \text{False positive}}.$$

Being true to form and correct is a fact. It provides the notion of accurately foreseen occurrences. The percentage of true positives from all positives is used to calculate precision.

Recall

Information that is important or valuable is the focus of the recall. The amount of pertinent data that is extracted from any machine learning method is measured by recall.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True positive} + \text{False Negative}}.$$

F1-score

To determine the test's accuracy, the F1-score is utilised. It is calculated precisely and recallably.

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

MODEL COMPARISON

This section presents a comparison of various machine learning models trained on the CTU-13 dataset to detect botnet traffic. The models evaluated include Support Vector Machine (SVM), Long Short-Term Memory (LSTM) neural networks, Logistic Regression, Random Forest, and two hybrid models combining these methods (SVM + LSTM, Random Forest + Logistic Regression). The goal is to identify the model that performs best in terms of accuracy and robustness for this specific dataset.

Models Evaluated

1. **Support Vector Machine (SVM):**

Accuracy: 89.51%

SVM is a robust classifier that is effective in high-dimensional spaces. It performed reasonably well on the CTU-13 dataset, achieving an accuracy of 89.51%. However, compared to other models, its performance was lower, likely due to the complexity and scale of the data.

2. Long Short-Term Memory (LSTM):

Accuracy: 96.77%

LSTM networks are a type of recurrent neural network (RNN) well-suited for sequential data. This model performed significantly better than SVM, achieving an accuracy of 96.77%. The LSTM model's ability to capture dependencies over sequences made it more effective in detecting patterns in the dataset.

3. Logistic Regression:

Accuracy: 85.17%

Logistic Regression, while being a simpler model, showed the lowest accuracy among the models tested. Its performance indicates that, while useful, it might not be complex enough to capture the intricate patterns in the CTU-13 dataset.

4. Random Forest:

Accuracy: 99.62%

The Random Forest model, which is an ensemble of decision trees, showed outstanding performance with an accuracy of 99.62%. Its ability to handle a large number of features and prevent overfitting made it the top performer among the individual models.

5. Hybrid Model: SVM + LSTM:

Accuracy: 94.90%

The hybrid model combining SVM and LSTM aimed to leverage the strengths of both models. While it outperformed the standalone SVM model, it did not surpass the individual LSTM model, suggesting that the hybrid approach was not as effective for this dataset.

6. Hybrid Model: Random Forest + Logistic Regression:

Accuracy: 99.60%

This hybrid model combined the robustness of Random Forest with the interpretability of Logistic Regression. While its performance was slightly lower than the standalone Random Forest model, it still achieved a very high accuracy of 99.60%.

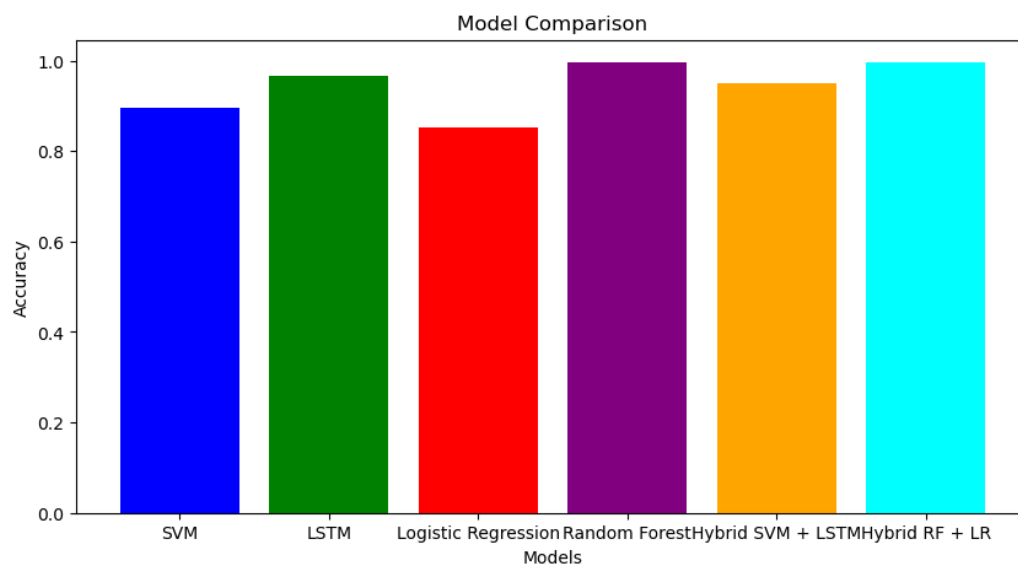


Figure 28: Comparison of Model Accuracies Across Different Algorithms

Random Forest emerged as the best-performing model with an accuracy of **99.62%**, slightly outperforming the hybrid Random Forest + Logistic Regression model. This indicates that the ensemble method of Random Forest is highly effective for the CTU-13 dataset, likely due to its ability to handle complex interactions between features.

The **LSTM model** also showed strong performance with **96.77%** accuracy, highlighting its suitability for sequential data but falling short compared to Random Forest. The hybrid models, particularly the **SVM + LSTM hybrid**, did not meet the expected improvement over individual models, suggesting that in this case, combining these models did not provide additional benefits.

Logistic Regression performed the lowest, reinforcing that more complex models are necessary to capture the complexities of the dataset.

CHAPTER 4: DISCUSSION, LIMITATION, CONCLUSION AND FUTURE STUDIES

DISCUSSION

Why Random Forest Performs the Best?

Random Forest emerged as the top-performing model in the analysis of the CTU-13 dataset, achieving an accuracy of 99.62%. This model's superior performance can be attributed to several key factors inherent to the Random Forest algorithm, which make it particularly well-suited for the type of data and classification task at hand.

In this project, the Random Forest model demonstrated the best performance among all tested models, achieving an impressive accuracy of 99.62% on the CTU-13 dataset. This success can be attributed to several key factors inherent in the Random Forest algorithm that align well with the nature of the dataset and the classification task.

Random Forest's ensemble approach, which involves creating multiple decision trees and aggregating their predictions, effectively captures the complex interactions between features within the dataset. This characteristic is particularly important given the high dimensionality of the CTU-13 dataset. By averaging the outputs of various trees, the model reduces variance and minimizes the risk of overfitting, which is a common challenge in single-model approaches like SVM and Logistic Regression.

Moreover, Random Forest is robust to noise and can handle both continuous and categorical data efficiently. Its ability to rank feature importance allows the model to focus on the most significant indicators of botnet traffic, enhancing its predictive accuracy. The ensemble nature also helps the model generalize well to unseen data, which is crucial for detecting malicious traffic in diverse network environments.

In comparison to the other models evaluated, such as SVM, LSTM, and Logistic Regression, Random Forest consistently outperformed them by leveraging its strength in managing complex, high-dimensional data and providing robust predictions, making it the most effective model for this task.

LIMITATIONS OF HYBRID MODEL

The hybrid models in this project, combining SVM with LSTM and Random Forest with Logistic Regression, did not outperform the standalone Random Forest model, and several factors could explain this limitation.

Firstly, hybrid models introduce additional complexity, requiring the stacking of predictions from base models, which can lead to a loss of information or the amplification of errors from individual models. If the base models do not complement each other well, the hybrid model might struggle to learn meaningful patterns from the combined outputs. For example, the SVM might capture linear relationships effectively, while LSTM excels at sequence data, but if these strengths do not align with the specific characteristics of the CTU-13 dataset, the hybrid model might not improve over simpler models like Random Forest.

Moreover, the Random Forest model's inherent ability to handle a wide variety of data types and its ensemble approach might inherently capture the complexities of the dataset better than a hybrid model. Random Forest also benefits from feature importance analysis, helping it focus on the most relevant features, whereas hybrid models might not leverage this advantage effectively.

Finally, the hybrid model's complexity can lead to overfitting, especially if the combined model fails to generalize well to new data. This overfitting might explain why the hybrid model did not perform as well as expected, highlighting the importance of selecting base models that truly complement each other in hybrid approaches.

Unexpected Outcome:

Initially, it was hypothesized that hybrid models would outperform individual models due to their ability to combine strengths and mitigate weaknesses. Surprisingly, the Random Forest model alone achieved the highest accuracy, even surpassing the hybrid models. This unexpected result underscores the exceptional capability of the Random Forest model in handling the dataset and suggests that it may offer the best balance of accuracy and simplicity for cyber threat detection.

CONCLUSION

This project explored various machine learning models, including SVM, LSTM, Logistic Regression, Random Forest, and hybrid models combining these approaches, to classify and predict outcomes within the CTU-13 dataset. Through extensive experimentation, it was observed that while hybrid models offered a sophisticated method of integrating different learning paradigms, they did not outperform the Random Forest model. The Random Forest model emerged as the best performer, likely due to its ability to handle a diverse range of

features, its robustness against overfitting, and its inherent capacity to capture complex, non-linear relationships within the data.

The results highlight that, despite the potential of hybrid models, simplicity and model alignment with the data's nature can yield better outcomes. The Random Forest's ability to prioritize relevant features and aggregate decisions from multiple trees provided a significant advantage, leading to superior accuracy and reliability in predictions. This project underscores the importance of carefully selecting and tuning models based on the specific characteristics of the dataset.

FUTURE STUDIES

While this study has shown the effectiveness of the Random Forest model in detecting botnet activities, there are several areas for future research that could further enhance the field of cyber threat detection:

Future work could focus on refining hybrid models by experimenting with different combinations of algorithms, especially those that could better complement each other. Techniques such as stacking or blending could be explored to see if they can address the limitations observed in this study. The CTU-13 dataset provided a solid foundation for this research, but future studies could benefit from incorporating more recent or diverse datasets to test the models' generalizability. Exploring datasets that include newer types of cyber threats could also provide additional insights and improve detection capabilities. Given the success of Random Forest, other ensemble methods, such as Gradient Boosting Machines (GBM) or XGBoost, could be explored for their potential to offer even better performance. These methods could provide a different approach to handling the complexities of cyber threat detection. Future research could also investigate the implementation of these models in real-time detection systems, evaluating their performance in live environments. This could involve optimizing models for speed and efficiency, ensuring they can provide accurate predictions under the time constraints of real-world applications. As cyber threats evolve, it is crucial to develop models that can adapt to new and unknown threats. Future work could explore how machine learning models can be designed to be more adaptive, potentially through continuous learning frameworks that update models in response to new data.

By addressing these areas, future research can build on the findings of this study, potentially leading to more robust, accurate, and versatile cyber threat detection systems.

CHAPTER 5: BIBLIOGRAPHY

CTU University, *The CTU-13 Dataset*. Available at: (Accessed: 24 August 2024).

Alothman, B., 2018. Similarity based instance transfer learning for botnet detection. *International Journal of Intelligent Computing Research (IJICR)*, 9, pp.880–889. [CrossRef]

Silva, S.S., Silva, R.M., Pinto, R.C. & Salles, R.M., 2013. Botnets: A survey. *Computer Networks*, 57, pp.378–403. [CrossRef]

Limarunothai, R. & Munlin, M.A., 2015. Trends and challenges of botnet architectures and detection techniques. *Journal of Information Science and Technology*, 5, pp.51–57.

Ghafir, I., Svoboda, J. & Prenosil, V., 2015. A survey on botnet command and control traffic detection. *International Journal of Advanced Computer Networking and Security*, 5, pp.75–80.

Haddadi, F., Morgan, J., Gomes Filho, E. & Zincir-Heywood, A.N., 2014. Botnet behaviour analysis using IP flows: With HTTP filters using classifiers. In *Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, BC, Canada, 13–16 May 2014, pp.7–12.

An introduction to the cyber threat environment. Available at: <https://www.cyber.gc.ca/en/guidance/introduction-cyber-threat-environment> (Accessed: 24 August 2024).

CrowdStrike, *Threat Detection, Investigation, and Response (TDIR)*. Available at: <https://www.crowdstrike.com/cybersecurity-101/threat-intelligence/threat-detection-investigation-response-tdir/> (Accessed: 24 August 2024).

Rapid7, *Threat Detection*. Available at: <https://www.rapid7.com/fundamentals/threat-detection/> (Accessed: 24 August 2024)

. IBM, *Cyber Threat Types*. Available at: <https://www.ibm.com/think/topics/cyberthreats-types> (Accessed: 24 August 2024).

Han, J., Kamber, M. & Pei, J., 2011. *Data mining concepts and techniques* (3rd ed.). Morgan Kaufmann Series in Data Management Systems.

Cortes, C. and Vapnik, V. (1995) 'Support-vector networks', *Machine Learning*, 20(3), pp. 273-297.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436-444.

Wolpert, D.H. (1992) 'Stacked generalization', *Neural Networks*, 5(2), pp. 241-259.

Gomes, R., Chowdhury, M.M. and Rifat, N. (2022) 'Cybersecurity threats and their mitigation approaches using machine learning—A review', *Journal of Cybersecurity and Privacy*, 2(3), pp. 527-555.

Arjunan, S. (2022) 'Deep reinforcement learning for cybersecurity threat detection and protection: A review', *arXiv preprint*, arXiv:2206.02733.

Guyon, I. and Elisseeff, A. (2003) 'An introduction to variable and feature selection', *Journal of Machine Learning Research*, 3, pp. 1157-1182.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, 12, pp. 2825-2830.

Silva, S.S., Silva, R.M., Pinto, R.C. and Salles, R.M. (2013) 'Botnets: A survey', *Computer Networks*, 57, pp. 378-403. [CrossRef]

Limarunothai, R. and Munlin, M.A. (2015) 'Trends and challenges of botnet architectures and detection techniques', *Journal of Information Science and Technology*, 5, pp. 51-57.

Muhammad, A., Asad, M. and Rehman Javed, A. (2020) 'Robust early-stage botnet detection using machine learning', *2020 International Conference on Cyber Warfare and Security (ICCWS)*, IEEE, pp. 1-6. Available at: <https://doi.org/10.1109/ICCWS48432.2020.9292395>.

Garcia, S., Grill, M., Stiborek, J. and Zunino, A. (2014) 'An empirical comparison of botnet detection methods', *Computers & Security*, 45, pp. 100-123.

Bilge, L., Balzarotti, D., Robertson, W., Kirda, E. and Kruegel, C. (2012) 'Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis', *Proceedings of the 28th Annual Computer Security Applications Conference*, ACM, pp. 129-138.

Tariq, F. and Baig, S. (2017) 'Machine learning based botnet detection in software defined networks', *International Journal of Security and Its Applications*, 11(11), pp. 1-12.

da Silva, A.S., Machado, C.C., Bisol, R.V., Granville, L.Z. and Schaeffer-Filho, A. (2015) 'Identification and selection of flow features for accurate traffic classification in SDN', *Network Computing and Applications (NCA)*, *2015 IEEE 14th International Symposium on*, pp. 134-141.

Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J. (2005) *Practical machine learning tools and techniques* (2nd ed.). Morgan Kaufmann.

Dua, S. and Du, X. (2016) *Data Mining and Machine Learning in Cybersecurity*. Boca Raton, FL: CRC Press.

Ester, M., Kriegel, H.P., Sander, J. and Xu, X. (1996) 'A density-based algorithm for discovering clusters in large spatial databases with noise', *In Proceedings of the KDD-96*, Portland, Oregon, pp. 226-231.

□ Inokuchi, A., Washio, T. and Motoda, H. (2008) 'An apriori-based algorithm for mining frequent substructures from graph data', *First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008)*, Adelaide, Australia, IEEE, pp. 10-16.

Li, Z., Zhang, A., Lei, J. and Wang, L. (2007) 'Real-time correlation of network security alerts', *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'07)*, Hong Kong, China, IEEE, pp. 73-80.

Blowers, M. and Williams, J. (2014) 'Machine learning applied to cyber operations', *Network Science and Cybersecurity*, Springer, pp. 155-175.

Sequeira, K. and Zaki, M. (2002) 'Admit: Anomaly-based data mining for intrusions', *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, AB, Canada, pp. 386-395.

Zhengbing, H., Zhitang, L. and Junqi, W. (2008) 'A novel network intrusion detection system (NIDS) based on signatures search of data mining', *Proceedings of the First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008)*, Adelaide, Australia, IEEE, pp. 10-16.

Şahin, D.Ö. and Demirci, S. (2020) 'Spam filtering with KNN: Investigation of the effect of k value on classification performance', *Proceedings of the 2020 28th Signal Processing and Communications Applications Conference (SIU)*, Gaziantep, Turkey, IEEE, pp. 1-4.

Sarker, I.H. (2019) 'Context-aware rule learning from smartphone data: Survey, challenges and future directions', *Journal of Big Data*, 6(1), pp. 1-25. [CrossRef]

MacQueen, J. (1965) 'Some methods for classification and analysis of multivariate observations', *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Oakland, CA, USA, pp. 281-297.

Ricci, F., Rokach, L. and Shapira, B. (2011) 'Introduction to recommender systems handbook', *Recommender Systems Handbook*, Springer, pp. 1-35.

Sneath, P.H. (1957) 'The application of computers to taxonomy', *Microbiology*, 17(1), pp. 201-226. [CrossRef]

Sørensen, T.A. (1948) 'A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons', *Biologiske Skrifter*, 5, pp. 1-34.