Maximizing Security by Integrating OPSEC and AI for Enhanced Defense Strategies

by

Raiyan Ferdous
21101127
Mamsad Ibn Yeahia
21101163
Munib Fahmid Rafeed
24341128
Nafiul Majid
21101169

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
December 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Raiyan Ferdous 21101127	Mamsad Ibn Yeahia 21101163
Munib Fahmid Rafeed 24341128	Nafiul Majid 21101169

Approval

The thesis/project titled "Maximizing Security by Integrating OPSEC and AI for Enhanced Defense Strategies" submitted by

1. Raiyan Ferdous (21101127)
2. Mamsad Ibn Yeahia (21101163)
3. Munib Fahmid Rafeed (24341128)
4. Nafiul Majid (21101169)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December 16, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Operational Security (OPSEC) is a risk and security management process and approach that first classifies information. There are five steps in OPSEC and the most important step is to identify and analyze potential threats and vulnerabilities. Analyzing threats and vulnerabilities can be done using manual penetration and VAPT where testers simulate real-world attacks the system may face in the future. Moreover, implementing improved machine learning algorithms can detect threats more robustly. By incorporating artificial intelligence with OPSEC, we can mitigate most of these drawbacks. This research identifies four threats; DDoS, Data Breach, Malware, and Phishing and detects them with particular ML/AI models and improves the performance of those models by optimizing. Moreover, this research implemented Explainable-AI (SHAP) in order to easily explain and interpret the models.

Keywords: OPSEC, VAPT, Threat Modeling, Vulnerability, Data Breach, Detection, Cyberattack, Penetration Testing, DDoS, Phishing, Malware. Ex-AI.

Table of Contents

Declaration i

Approval ii

Abstract iii

Table of Contents iv

1 Introduction 1
- 1.1 Problem Statement . . . 2
- 1.2 Research Objectives . . . 4

2 Literature Review 5
- 2.1 Related Work . . . 5

3 Working Plan 10
- 3.1 Threat Definition . . . 12
  - 3.1.1 Definition of DDoS Attack . . . 12
  - 3.1.2 DDoS Threats . . . 12
  - 3.1.3 Overview of Phishing Attack Definition: . . . 13
  - 3.1.4 Data Breach . . . 14
  - 3.1.5 Malware . . . 15
- 3.2 Model Defination . . . 16
  - 3.2.1 Random Forest . . . 16
  - 3.2.2 Support Vector Machine (SVM) . . . 17
  - 3.2.3 Logistic Regression . . . 17
  - 3.2.4 K-Nearest Neighbor (KNN) . . . 17
  - 3.2.5 Neural Network . . . 17
  - 3.2.6 Decision Tree . . . 18

4 Dataset Description 19
- 4.1 Dataset for DDoS Detection . . . 19
  - 4.1.1 Data analysis of DDoS dataset . . . 19
  - 4.1.2 Data Preprocessing of DDoS Attack SDN Dataset . . . 21
- 4.2 Dataset for Data Breach . . . 26
  - 4.2.1 Data Analysis . . . 26
  - 4.2.2 Data Preprocessing . . . 28
- 4.3 Dataset for Malware . . . 31
  - 4.3.1 Dataset Description . . . 31

4.3.2 Data Preprocessing of Malware Dataset 33
4.4 Dataset for Phishing Attack 34
4.4.1 Data Analysis 34
4.4.2 Data Preprocessing of Phishing Attack CSV Dataset 35

5 Model Implementation and Results Analysis **39**
5.1 Results 39
5.1.1 DDoS Detection Results 39
5.1.2 Data Breach Results 43
5.1.3 Malware Results 47
5.1.4 Phishing Attack Results 51
5.2 Discussion 55
5.3 Constraints 56

6 Conclusion **57**
6.1 Future Works 58

Bibliography **62**

Chapter 1

Introduction

In today's interconnected world with various cyber threats, the inclusion of Operational security OPSEC with Artificial Intelligence (AI) is a state-of-the-art method in cyberspace that demonstrates the future potential to enhance defense strategies. In the age of ever-evolving cyber threats, traditional risk and vulnerability management approaches often fall short of effectively mitigating risks. Traditional methods of vulnerability assessments such as manual penetration testing and using semi-automatic software come with multiple drawbacks. Not only are these methods time-consuming but they also bring various issues such as; scalability issues, human error, high rate of false positives and negatives, etc. Therefore, the synergistic linkages between OPSEC and AI propose a comprehensive framework to maximize security through their integration.

Operational Security (OPSEC) can be defined as a systematic procedure that seeks to recognize and secure classified information from attackers. The US military first came up with OPSEC but it has been embraced by different other industries to safeguard critical data through assessing and controlling possible threats. The major aim of OPSEC is to predict and prevent possible intrusions using careful mapping out of security protocols. However, traditional OPSEC measures are limited in effectiveness against flexible online menaces due to their static nature [15].

In this world of globalization, incorporating OPSEC and AI is a forceful way to improve defense strategies. AI has predictive and analytical abilities that can significantly enhance traditional OPSEC tools for better performance. This integration aims at dealing with various security challenges such as data breaches, cyber espionage as well as other forms of malicious activities. Vulnerability assessment and analysis of cyber threats is one of the major security challenges that exist in modern defense strategies. These may involve anything from malware and phishing attacks to more complex intrusions or distributed denial-of-service (DDoS) attacks. Traditional OPSEC measures are different from AI since they can also provide real-time threat and vulnerability analysis in addition to automated responses for effective responses to assess various kinds of cyber attacks [43].

On 27th May 2023, the MOVEit attack commenced and it was a cyberattack on MOVEit, a secure file transfer software. Hackers possibly linked to the Clop ransomware group exploited a zero-day susceptibility in MOVEit file transfer software.

Zero-day vulnerability refers to an unprotected gap in security that has not been identified by the software maker itself. However, SQL commands were inserted into the database by hackers. The attackers stole sensitive information from MOVEit customers and multinational companies like the BBC and British Airways. This resulted in data breaches, with stolen data potentially used for extortion or sold on the black market. Moreover, it affected their clients whose data was also compromised. This breach illustrates how lax OPSEC can cascade through supply chains to impact more than just breached organizations' customers [44].

Cyber defense systems driven by AI and ML will be significant in addressing the ever-increasing quantity and complexity of threats, their evolving characteristics, and the necessity for swift and greatly automated responses to threats. Defense systems powered by AI and ML can not only analyze large sets of data but can also recognize anomalies and dubious patterns promptly. Furthermore, large-scale cyber attacks can be prevented by providing automatic updates to software that already exists based on complex real-time analysis by ML or AI [24].

There are several ways to incorporate AI with OPSEC. One such approach is using ML where predictive models for vulnerability classification, clustering, and ranking, and the most commonly used algorithms are Naive Bayes, Random Forests, Logistic Regression and Support Vector Machines. Web application security vulnerabilities such as SQL injection and denial of service are also found using ML approaches [10]. Here, the primary focus is on creating attack strategies that can assess the underlying system's security [6]. According to the National Institute of Standards and Technology (NIST), attack graphs are used to represent a multiple set of vulnerabilities that can be used to launch an attack [8]. Expert Systems, software that mimics human behavior, are also used to detect vulnerabilities [2]. Furthermore, fuzzy logic systems are used as well for improving the vulnerability assessment process [1]. As technology is improving day by day, computers and systems are being subjected to various types of attacks, thus, by incorporating AI methodologies with OPSEC we can maximize security strategies. Hence, finding the best suited AI model for each of the common vulnerabilities and using them accordingly will provide the most effectiveness in terms of detecting vulnerabilities in systems.

1.1 Problem Statement

Penetration testing also known as Pen testing is a permitted simulated digital attack conducted on a computing system to assess its security. Pen testing aims to find vulnerabilities in systems that attackers could potentially use to exploit and assess the consequences of those exploits. This aids in offering feasible suggestions to improve the security of the system. Penetration testers identify and illustrate the financial impacts of system vulnerabilities using the same tools, strategies, and procedures as attackers. Regular attacks used by attackers can be classified as Probing, Remote to user (R2L) User to root (U2R), and Denial Of Service (DoS) [7]. In probing, a network of computers is scanned by an attacker to obtain data or to identify known vulnerabilities. An attacker can utilize this data to search for exploits with the map of the computers and services that are available on the network. U2R approach is

when an attacker can obtain root access to a system by first gaining entry to a regular user account and then exploiting system vulnerabilities. DoS method is when a hacker overloads several computational resources and so it cannot operate authorized requests. R2L is the method where to get local entry as a user of a remote machine, an attacker without an account utilizes a vulnerability and sends packets to that machine across a network. The goal of these attacks is to steal, ruin, or read data. There are three levels of pen testing: Grey box testing, Black box testing and White box testing: In black box testing, the target system's internal structure is unknown to the tester. This approach imitates an external attacker trying to exploit the system from the outside. And in Grey box testing, the information on the system's inner structure is limited. This approach has features of both black box and white box testing. And finally, in White box testing, the system's inner structure is fully known which includes the source code and algorithms. This method is more in-depth and seeks to find vulnerabilities that might not be visible from the outside.

In the real world, we have seen that Artificial intelligence and machine learning have contributed a lot to penetration Testing problems. To develop AI-driven tools for any automated penetration testing there can be some problems with tools such as ensuring these tools can adapt to new vulnerabilities and attack vectors while maintaining high-low accuracy and low false-positive rates [12]. In [3], Pen testing can pose a problem in evaluating security in multi-tenant cloud environments. As it is difficult to simulate attacks in a way that respects other tenants' privacy and resources and ensures comprehensive coverage without disrupting services. Executing pen testing in blockchain technology can also be quite a challenging task as it will be hard to identify unique vulnerabilities associated with consensus algorithms and smart contract execution. And so this causes difficulty in assessing the security of blockchain implementations and smart contracts. Pen testing and red team automation are both complementary techniques in cybersecurity testing. Red team automation simulates advanced attacks and evaluates the system's overall defensive capabilities continuously and comprehensively. However, there will be difficulty in automating red team activities to improve efficiency and coverage. Ensuring that the automated tools can mimic the creativity and adaptability of human attackers will be quite the challenge. Simulating realistic Social Engineering Attacks such as phishing, pretexting, baiting, etc., is challenging too. Balancing realism with ethical considerations and ensuring employee consent and awareness is difficult. There can also be problems in automating the reconnaissance phase of penetration testing. As, collecting and analyzing large amounts of data while avoiding detection and maintaining efficiency will be challenging. Simulating Advanced Persistent Threats (APTs) to test long-term, stealthy attack scenarios can cause problems as well [16]. Because it is challenging to create realistic, prolonged attack simulations that accurately reflect the tactics and techniques of APT groups. Finally, there can also be issues in navigating the legal and ethical boundaries of penetration testing [26]. Ensuring compliance with laws and regulations while effectively testing security measures is quite challenging.

1.2 Research Objectives

Our research aims to explore various machine learning models for detecting vulnerabilities, improve models with low accuracies, and determine the most suitable model for detecting potential threats. The objectives of this research are:

1. In our work, an optimized ML model is proposed for each threat, which is the best for detecting that vulnerability in all aspects. Our aim here is to increase the efficacy and overall effectiveness of penetration testing methods.

2. Furthermore, the aim is to reduce human error and provide some actionable insights for the result.

3. At first, various machine learning models are employed to identify various vulnerabilities. After that, automated exploitation techniques are deployed to test the strength of the identified vulnerabilities.

4. Moreover, we need to improvise models with low effectiveness in terms of detecting particular threats.

5. Lastly, we evaluate the performance of each framework through real-world testing scenarios and then compare it to traditional penetration testing methods in accuracy, effectiveness, efficiency, speed and lastly comprehensiveness. At last, we will propose one ML model for each vulnerability, which is most appropriate for identifying systems that are at that risk.

Chapter 2

Literature Review

2.1 Related Work

Dawson et al. [5] (2017) portrayed how they use some combinations of different testing methods to improve security-based risk assessment in their test-driven threat modeling, STRIDE, CAPEC etc. Penetration testing techniques are crucial in addressing the growing threats in cybersecurity. Also it emphasizes the importance of combining traditional methods such as rule analysis and fuzzing with advanced techniques such as threat classification This integrated approach brings detailed abuse profiles and provides greater specificity than traditional. The study emphasizes that such integration improves the risk assessment accuracy and also it helps for bettering resource allocation to mitigate software vulnerabilities.

In their research, Arfaj et al. [27] (2021) explore both conventional and unconventional penetration testing techniques to enhance cybersecurity measures. The growing sophistication of cyberattacks has underscored the need for advanced penetration testing methods to safeguard organizations and government entities from significant financial and confidential losses. The authors emphasize the inadequacy of conventional methods in addressing the ever-evolving and dynamic nature of cyber threats, advocating for the adoption of more advanced trying out mechanisms. Their examination proposes a penetration of Wi-Fi by using approach based on Kali Linux, structured into 4 stages: preparation, information amassing, assault simulation, and reporting. This methodology consists of tracking, scanning, capturing, reading, and password cracking techniques, demonstrating its efficacy in enhancing the security evaluation of wireless conversation systems. The findings highlight the necessity of integrating both manual and automated penetration testing techniques to effectively address emerging security threats, thereby justifying the continued relevance and evolution of penetration testing in cybersecurity.

Altulaihan et al. [33] (2023) examined the features of several popular web penetration testing tools. The research also showcased the advantages of Automated pen testing, one main advantage is that it saves time. Web application vulnerabilities are also covered in the research, along with mitigation strategies. Some of the vulnerabilities are: Injection, Security Misconfiguration, Broken Access Control, Server-Side Request Forgery (SSRF), Cryptographic Failure, Failures of Security Logging and Monitoring, Failures in Identification and Authentication, Outdated and Vulnerable

components and Unsafe Design. The authors gave a detailed comparison between 7 open source penetration testing tools. They are: OWASP ZAP, IronWASP, W3af, Wapiti, Acunetix, Netsparker, and Vega. Different web penetration tools have different features and so the authors recommend using the tools which have the features we require. The authors state that although web cybersecurity is becoming more and more popular, not much study has been done on web penetration testing tools. The authors therefore recommend conducting additional study on the application of automated approaches to improve web privacy and security.

McKinnel et al. [13] (2019) in their review compared the results of different types of AI disciplines used in pen testing vulnerability assessment. Independent variables (IV) are used to evaluate the suggested systems capabilities. The review demonstrates that the Local search for Planning Graphs (LPG-td) planner is inferior in application to the FIDIUS's Contingent Fast Forward (cFF) planning procedure. Pruning technique optimizes the algorithm decision making procedure. When dealing with metrics that increase exponentially, optimisation is essential. Scalability is crucial when testing Partially Observable Markov Decision Process (POMDPs) within penetration testing. Shorter attack pathways are created when using Fast Forward planning (FF). Training period increases productivity and so has a significant impact on assessing the solution's efficacy. Using both Fast-Forward planning (+FF) and Planning Domain Definition Language (PDDL) gives great results and is better than those with MDP or POMDP models. The author suggests developing systems that are realistic and have attributes that are common to many real world implementations. The authors believe that as systems in society become more intelligent and advanced, AI will play a bigger role in vulnerability assessment, thus we must stay up to date with technological advancements.

Hassan [28] (2022), proposed an approach with the implementation of VAPT (Vulnerability Assessment and Penetration Testing) to maximize web security by integrating Artificial Intelligence. The research emphasizes the fact that VAPT involves VA (Vulnerability Assessment); which scans for potential flaws in the security detecting the vulnerability in the system and PT (Penetration Testing) exploits these flaws and tries to break the defense system. PT or Pen-test is an authorized simulated cyber attack with a goal of testing the system's defense strength. Nessus and OpenVAS are automated tools that are underlined for vulnerability assessments. However, none of these tools can identify all security risks and vulnerabilities. Furthermore, Manual testing can be thorough but it is labor intensive and creates the chance of human error. XSS (Cross-site Scripting) and Sql Injection are sort of vulnerabilities that have been exploited since the late 1990s and XSS has ranked third in the web application vulnerabilities list. Functional Pen Testing; Grey Box Testing and White Box Testing are proactive and systematic methods of assessing vulnerabilities. However, due to the advancement of the internet and Offensive-AI assessing vulnerabilities is becoming harder and ineffective. The integration of AI in VAPT can effectively analyze large and complicated datasets by implementing AI algorithms in order to detect potential security threats and prioritize vulnerabilities based on risk levels. Therefore, an AI-driven approach with the help of automated and manual tools can increase the effectiveness of vulnerable systems.

Meliopoulos et al. [4] (2015) proposed a new infrastructure that improves the power grid security by enabling parallel use of operational security and cyber security. The technique foundations are interception of commands and quick authentication from cyber security and operational reliability's perspective. As the commands arrive the relay and its control circuits, this infrastructure has the capability to detect, authenticate, and stop the commands. Since every control works through a relay, this method offers complete coverage. There's also an open source instantaneous network monitoring solution to gather and decipher network traffic. The suggested approach is deemed immune to a byzantine type assault because the dynamic state estimator at the substation level serves as its foundation. The proposed system collects data from substation devices, executes dynamic state estimation based security, using this data for the purposes of operational reliability and cybersecurity simultaneously. The approach produces a real time model with very low latency. The simulation runs faster than real time because a local model is used. The approach provides dynamic, real time semantic awareness. The system has the ability to anticipate and guard against future vulnerabilities. As a result, the power grid can be protected against both conventional network attacks and non conventional attacks. Therefore the research shows how simultaneous cybersecurity and Operational security can significantly improve the power grid security.

Sarker [25] (2021) emphasized on AI techniques such as Machine Learning (ML), Deep Learning (DL), Natural Language Processing (NLP), Knowledge Representation and Reasoning (KRR), and rule-based Expert Systems (ES) to intelligently tackle present cybersecurity issues. He mentioned that ML, especially neural network-based deep learning, AI's crucial aspect, can be utilized for constructing fruitful security models using the existing historical cybersecurity data. In this approach, supervised learning techniques such as Navies Bayes, K-nearest neighbors, Stochastic Gradient Descent etc. are used when target classes of attack-anomaly are bound to reach from a particular input set. Unsuper-vised learning is used to identify patterns from unlabeled data, where clustering is used to uncover the hidden patterns with clustering algorithms such as K-medoids, CLARA etc. He further mentioned NLP-based modeling for cyber-attack predictions and malicious domain identification with techniques such as semantic and syntactic analysis. Lastly, the author talked about the concept of knowledge-representation based security modeling and a cybersecurity expert system model based on rules that can function as a security expert in an intelligent framework to address complicated cybersecurity problems.

Zeadally et al. [20] stated that as cyber crimes are getting more sophisti-cated with the advancement in technology, traditional cybersecurity solutions are becoming insufficient in tackling them, whereas, AI techniques are showing promising results in countering the ever evolving cyberattacks. First of all, they mentioned that ML based cybersecurity solutions are being used not only to automate attack detections, but also to improve and evolve their capabilities with time, as ML based methods can handle large volumes of data and attributes.Secondly, the decision tree method provides an instinctive solution for detecting cybersecurity issues due to its ability to show the result of feature-based decision values which is necessary for classifying observed cybersecurity incidents as genuine attacks or not. In this method, after finding an effective series of rules, internet traffic can be classified by intrusion

detection systems in real time. Furthermore, the K-Nearest Neighbor method is suitable for incursion-detection systems because it can recognize zero-day assaults as its invisible classes by learning from novel traffic patterns. KNN method works well with data that can be represented by a model which measures its distance from other data. The authors also pointed out that Support Vector Machines (SVM) can be implemented in analyzing patterns of internet traffic consisting of several classes such as File Transfer Protocol (FTP) and HyperText Transfer Protocol (HTTP). SVM can be used in applications where simulation of attacks are done. The authors also discussed how Artificial Neural Networks (ANN) may be used in intrusion detection systems because of its adaptability to new communication channels. They also mentioned ANN's capability in detecting zero day attacks due to its learning from events in the recent-past. Moreover, ANN can be a well-suited detection system in cybersecurity related applications where at the time of an incident, the attack class can be labeled,thus, the detection system can learn from that incident. Lastly, the authors talked about visualization tools such as Self Organizing Map (SOM) that allows network operators in anomaly detection in network traffic such as zero day attacks and botnets.

Markevych and Dawson [36] (2023) talked about the potential of AI techniques such as Fuzzy Logic, Neural Networks (NN), Support Vector Machine (SVM) in cybersecurity applications to reduce false positives to enhance Intrusion Detection System's (IDS) capability and detection rate. They stated that AI based IDSs provide more advantages over traditional methods in terms of real time attack detection and response capacity, adaptability and recognizing attack patterns. IDS based on AI can adapt to evolving threats network characteristics over time which allows them to detect previously unseen attacks, providing a more powerful security against evolving cyber threats. Furthermore, by employing machine learning algorithms to identify patterns of malicious activity in massive quantities of network data, AI-based intrusion detection systems (IDS) may identify a wide range of cyberthreats, including Advanced Persistent Threats (APT), Zero Day Attacks, and more. Additionally, the use of advanced algorithms in AI based IDS, allows analyzing network traffic and malicious activities and responding to them in real time reducing the impact of cyberattacks. The authors also mentioned that ChatGpt and similar AI models can vastly reduce false positives by using advanced data analysis and correlation methods. These AI models can be retrained with the feedback from security experts allowing them to detect real threats. They showed that presently the most astounding examples of efficient AI driven IDS are seen in finance and banking sectors.

Khan et al. [11] (2018), introduced the developing techniques in vulnerability assessment for computer systems, highlighting its crucial importance in identifying and rectifying security loopholes before exploitation. The kinds of vulnerability assessment addressed by this study include manual, assistive, and fully automated. Furthermore, manual assessment depends on human expertise; it also requires considerable time and resources. On the contrary, assistive evaluation utilizes modern scanning tools but suffers from rigidity, maintenance issues and is not flexible. Complete automation of vulnerability assessment with the enhancement of AI can be time efficient and it can also be flexible. Data breaches can be detrimental and according

to a survey in 2016, usually 11 percent of customers face consequences due to potential data breaches. assessments using artificial intelligence (AI) are considered to be the best approach possible. Metasploit, AirCrack, Nmap, Burp Suite are some tools and frameworks that are available for VA, nevertheless, these tools differ in terms of accuracy. However, further investigation should focus on enhancing learning mechanisms as well as knowledge acquisition and representation in the case of automated methods. In conclusion, the paper stresses increasing research towards AI-driven VA's as well as a survey requirement that takes into account all recent developments.

Chapter 3

Working Plan

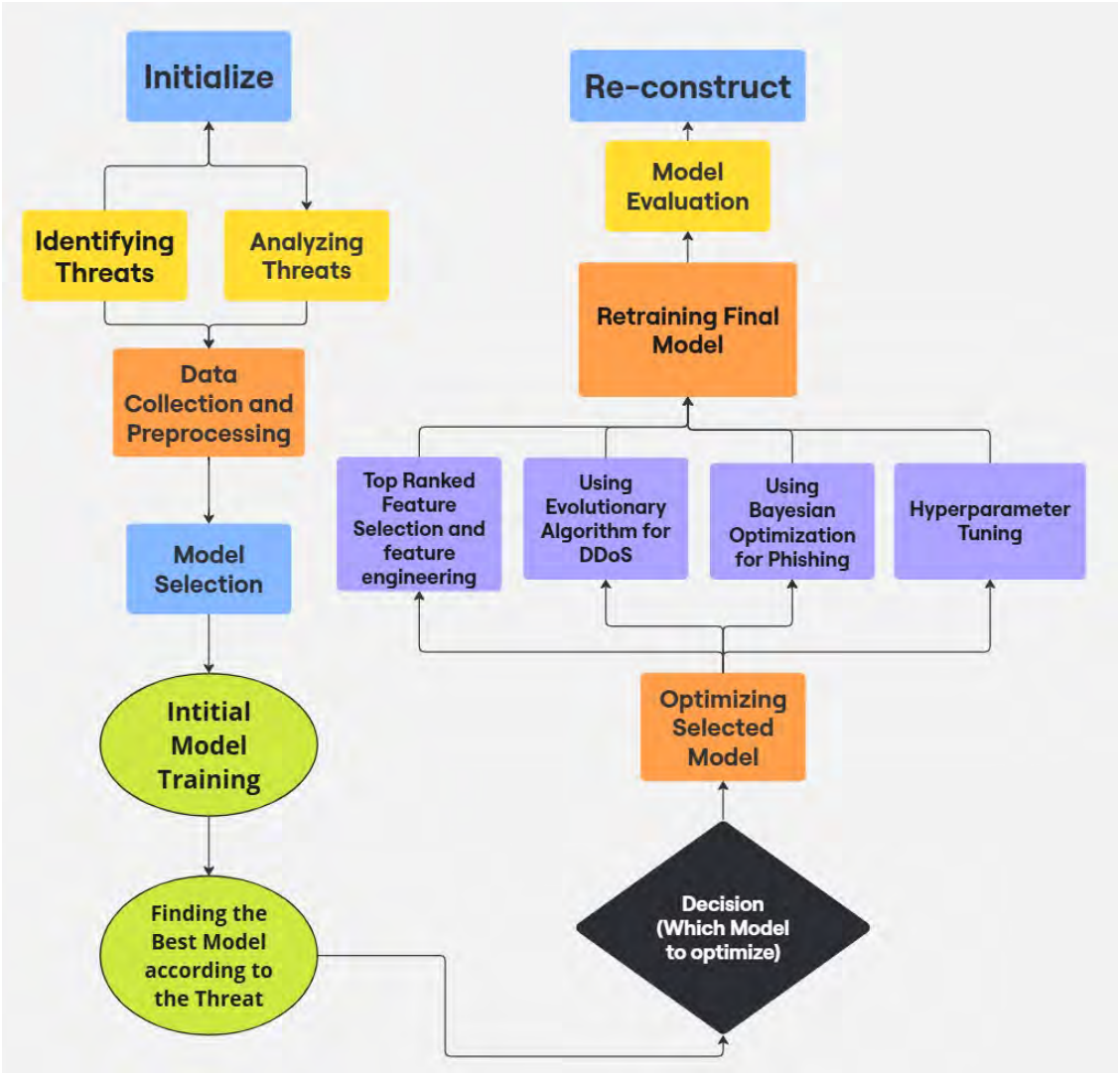

Figure 3.1: Work Plan

The figure 3.1 represents the process of improving four machine learning models.

1. Start Point: The process starts with the identification of the potential threats. This stage focuses on defining the scope of the problems, such as detecting the data

breach, DDoS, phishing and malware. This is also essential for understanding the nature of the threats and models with the objectives of the analysis.

2. Identifying and Analyzing Threats: When the potential threats are identified, they are ready to be analyzed to understand their characteristics. This includes studying patterns in data breaches and malware attacks. This analysis forms the basis for selecting the relevant datasets for the machine learning models.

3. Data Collection and Preprocessing: This step involves gathering the necessary data for all use cases. Along with resolving missing values, scaling numerical features, and encoding categorical variables, the raw data is also cleaned and preprocessed. By doing this, the data is guaranteed to be in a format that can be used to train machine learning models.

4. Model Selection: At this stage, appropriate models are selected based on their ability to handle data of high-dimensions and the distinguish between classes effectively.Moreover, other models were chosen for their effectiveness in binary classification tasks.

5. Initial Model Training: Both of these models start with default hyperparameters for their initial training. Metrics including accuracy, precision, recall, and F1-score are used to evaluate their performance. These baseline results reveal areas where improvements can be made.

6. Finding the Best Model According to the Threat: At this stage, after the initial training, there are information about which models are suitable for our implementation of improvements. Judging by the metrics and threats chararistics, its easier to understand which model is best according to our threat.

7. Decision: Deciding which model should be optimized further.

8. Optimizing the Selected Models with Multiple Approaches: Tuning hyperparameters is essential for improving a model's performance. A search strategy, such as GridSearchCV, is employed to investigate various combinations of hyperparameters. Feature engineering and selecting top features can also help a model optimize itself. Using evolutionary algorithms and Bayesian optimization also improves models performance.

9. Retrain Final Model: Once the optimal hyperparameters are identified, based on selecting the top features using the models, they are retrained using the complete training dataset. This step ensures that the models are fully optimized and prepared for the assessment of the particular test-set.

10. Model Evaluation: The final models undergo evaluation to assess their performance based on the test set. The results are compared to the baseline models to verify the improvements gained through hyperparameter tuning. Enhanced performance metrics and visualization methods, such as learning curves and using SHAP AI, further illustrate the effectiveness of the optimized models.

11. Reconstruction: Finally, the entire workflow is reviewed and refined as necessary. This ensures a systematic approach to developing robust models for detecting data breaches and malware.

3.1 Threat Definition

3.1.1 Definition of DDoS Attack

A Distributed Denial-of-Service (DDoS) attack is defined as an attempt to render a target system, including a server or network, inaccessible to legitimate users by overwhelming it with traffic from multiple machines. These attacks are particularly challenging to defend against due to their distributed nature, which entails a large number of compromised devices, commonly known as a botnet, that simultaneously send a massive volume of requests or data to the target, leading to the depletion of essential resources such as bandwidth, memory, or CPU.

3.1.2 DDoS Threats

DDoS (Distributed Denial-of-Service) attacks pose an important threat to the financial sector by overwhelming online services, such as banking and payment processing systems, with excessive traffic. These attacks can cause legitimate users to lose access to services, resulting in immediate financial losses due to disrupted transactions and reputational damage stemming from service outages. As institutions strive to maintain reliable operations, consumer trust may be severely affected, prompting clients to look for alternative solutions. The financial repercussions of restoring services and implementing strong DDoS mitigation strategies can further strain resources, emphasizing the necessity for financial organizations to invest in comprehensive security measures and incident response plans to defend against such threats.

DDoS Attack Patterns:

The Internet infrastructure, designed primarily for performance, is more susceptible to Distributed Denial-of-Service (DDoS) attacks. This design employs an end-to-end model, where the majority of intelligence resides at communication endpoints (hosts), while the intermediate network primarily functions to forward packets. Such a configuration allows attackers to exploit the network's high bandwidth to target a specific system. A notable vulnerability is that security is often self-contained; even well-secured systems can be compromised when other devices on the Internet are breached, leading to the formation of botnets. Attackers frequently exploit the limited resources of Internet service providers, such as bandwidth or CPU, by overwhelming them with traffic. The capability of many devices surpasses that of a few, allowing a botnet comprised of thousands of machines to considerably exceed the resources of the victim and flood the target with traffic. One major weakness exploited by attackers is the lack of accountability in IP address verification, enabling addresses to be fabricated and creating difficulty in tracing the origins of an attack.

This situation allows for attacks such as reverse attacks, where web requests from multiple compromised servers inundate the target. Besides, the decentralized nature of the Internet complicates the application of uniform security measures across all networks, resulting in the indiscriminate spread of DDoS attacks.
DDoS attacks pose an escalating threat to the financial sector, disrupting online services and resulting in considerable financial losses. These attacks inundate the networks of financial institutions with enormous traffic, leading to operational downtime and threatening sensitive data. The detail and magnitude of DDoS attacks are advancing, rendering traditional detection methods insufficient. Nonetheless, AI is becoming a formidable instrument in the fight against these attacks. AI-driven systems possess the capability to analyze extensive datasets in real-time, identify anomalies in network traffic, and detect potential threats more efficiently than conventional methods. In particular, machine learning algorithms excel at recognizing patterns in data, enabling them to anticipate and mitigate DDoS attacks before any damage occurs. Besides, AI can automate responses to such threats, thereby minimizing reaction times and enhancing overall security resilience (Al-Shareeda et al., 2021) [37]. As the financial sector increasingly relies on digital infrastructure, AI provides a proactive strategy for protecting these systems from DDoS attacks, ensuring greater security for both institutions and their customers.

3.1.3 Overview of Phishing Attack Definition:

This type of identity theft merges social engineering techniques with sophisticated attack vectors that lead to the compromise of sensitive financial or personal information of the targeted individual. Victims often fall prey to phishing attempts by being deceived into clicking on fraudulent URLs that redirect them to counterfeit websites designed to closely mimic legitimate sites, such as banking or e-commerce platforms. The primary objective is to mislead the individual into disclosing sensitive information, which may include passwords, credit card numbers, or other personal data.

Phishing Threats:

Phishing attacks present a important risk to the financial sector by targeting individuals and businesses in an effort to steal sensitive information, including passwords and account details. These schemes often trick victims into providing their personal data, leading to identity theft, financial fraud, and data breaches, finally resulting in substantial financial losses for both affected individuals and the organizations involved. The impact extends beyond immediate monetary loss; it may also include regulatory scrutiny and enduring reputational harm. As customers lose confidence in organizations that fail to safeguard their information, financial institutions may encounter increased operational costs associated with enhancing security measures and training staff. Consequently, strong security protocols and ongoing customer education are essential for effectively combating phishing threats.

Architecture of a Phishing Attack:

The architecture of phishing attacks typically consists of the following stages:

Attack Launch: The attacker initiates phishing by using email, messaging, or other methods to embed a fraudulent URL or attachment within the message. This URL often directs the victim to a counterfeit website designed to resemble a legitimate banking or e-commerce site.
User Interaction: The victim is deceived into clicking the link or opening the attachment, which may redirect them to a fraudulent webpage that appears legitimate. The page may also request sensitive information, such as their 'username/password' or financial details.
Data Harvesting: The phisher then collects the stolen facts without delay through the faux webpage or through malware inclusive of keyloggers downloaded with the attachment.
Spoofing Techniques: Spoofing techniques encompass URL obfuscation in order for the phishing website to present itself as legitimate-as an instance, it might consist of the name of the valid logo within the route of the URL at the same time as masking the domain in query.
Phishing attacks pose a consistent threat within the financial sector, aiming at both institutions and their clients. These attacks typically involve deceptive emails or websites that seek to deceive individuals into revealing sensitive information, including account credentials and personal data. Financial institutions are particularly susceptible to these attacks due to the sensitive nature of the information they manage. Traditional security measures, although beneficial, frequently fall short of keeping up with the growing complexity of phishing techniques. AI has the potential to greatly improve phishing detection by recognizing patterns and anomalies in email communications or web traffic that may go unnoticed by human operators. Machine learning algorithms can be developed to identify phishing attempts through the analysis of extensive datasets, enhancing detection accuracy over time (Madireddy et al., 2022)[29]. AI models are also capable of employing natural language processing to detect suspicious language in emails and automatically flag potential threats. In addition, AI-driven systems can function in real-time, assisting quicker responses and reducing the effects of phishing attacks. By using AI, the financial sector can fortify its defenses against phishing, finally safeguarding customer data and preserving trust.

3.1.4 Data Breach

When unauthorized people gain the access to some private, sensitive, protected, or confidential information, it is known as data breach. Data breaches enable illegal access to sensitive, private, or secured data. Data breaches might happen to anybody, including private citizens, large corporations, and governments. Without any proper protective measures, anybody may endanger others. Cyber-criminals might profit from stolen information by selling it or using it in conjunc-tion with a bigger enterprise. Sensitive information including bank account details, credit card numbers, personal health information, and email and social networking site login passwords are typically stolen in data breaches. In addition to monetary damages, such information breach can substantially impair a company's reputation with clients, customers, and employees, particularly in the financial industry. and employees, particularly in the financial industry.

The architecture of Data Breach:
A data breach might take place in the following ways:

- Through an accidental insider
- Through an insider who is malicious
- Through devices that are lost and/or stolen
- Through outsider criminals

Some notable methods applied by hackers to induce a data breach are: i) Malware, ii) Phishing, iii) Brute force attack.
There are various types of existing protective measures to safeguard ourselves against data breaches. According to Zhang et al. (2022) [31], the protective measures include secure access controls, administrative controls, security audit, encryption, intrusion detection through network monitoring and identity management systems. However, they also added that these existing protective measures have their own shortcomings and challenges. For instance, the growth rate of data is ever-increasing and due to this data explosion, it becomes challenging for the existing measures to protect this huge amount of data from data breaches. Implementing all of these protective measures becomes pretty expensive. Furthermore, often for separate environments, separate security patches are needed. Due to technological advancements, these existing methods often do not suffice in preventing data breaches. Last but not the least is human errors as human errors are the easiest way to instigate a data breach.
To deal with these challenges, incorporation of AI can be a viable solution. According to Ibrahim et al. (2020) [17], intelligence-driven architectures improve visibility of cyber risks, including data breaches, and reduce the threat environment. Implementing intelligence-driven architectures has issues that need to be addressed both methodologically and theoretically. However, they are a promising trend for future cyber protection technology. Integrating AI and ML into security systems allows for automation and orchestration, resulting in independent solutions capable of addressing modern cybersecurity risks like data breaches.

3.1.5 Malware

Malware attacks are the processes which involve deploying a malicious software (such as viruses, worms, trojans) on a certain network or device for obtaining sensitive data, to monitor people's, or even destroying the systems. Malware has the ability to replicate itself and infect a computer device without the user's knowledge or consent. According to Gaber et al. (2024), from 450 million in 2018 to 970 million in 2022, the overall amount of malware that targeted operating systems like Windows, Android, Mac, and Linux- increased more than two folds, and discovered that the average overall cost of data breaches across 17 countries was USD 4.35 million per occurrence. One of the solutions for Malware is Anti-Malware Software. Robust anti-malware software is installed on every system across the board including endpoint devices to identify, prevent, and eliminate any malicious software. Faruk et al.(2021) stated that techniques for detecting malware can be categorized into three groups: heuristic-based, anomaly-based, and signature-based[22]. Traditional approaches to anti-malware software based on signature identification face difficulties

with detecting new future malware (zero-day malware). According to Gaber et al. (2024), malware detection based on heuristics identifies suspicious activities by using a set of criteria developed by malware experts., but it can be error-prone[41]. Due to their malice, malware tends to develop quickly in order to outwit standard defences. The paper also stated that modern malware is so complex that signature-based heuristic detection can be readily bypassed as cyberattacks continue to get more sophisticated. The development of anti-malware systems may get help from the use of AI. According to Faruk et al. (2021), the use of AI technologies in malware detection and prevention needs to be increased in order to tackle the rise in intelligent malware in recent years [22]. Gaber et al. (2024) have demonstrated the high accuracy outcomes of various AI models, architectures, techniques, and parameters employed in malware detection. However, Gaber et al. (2024) have also demonstrated that employing AI for malware detection presents a number of difficulties [22]. Gaber et al. (2024) have shown the high accuracy results of Diverse types of AI used for malware detection, where many different models, architectures, approaches, and parameters are used. However, Gaber et al. (2024) have demonstrated that employing AI for malware detection presents a number of difficulties [41]. Understanding the basic phenomena of whether syntactic or semantic elements are absent from benign software but present in contemporary malware is crucial. According to Gaber et al. (2024), the quality and originality of the features that an AI model is trained on determines its accuracy and effectiveness [41]. Low-quality features from limited datasets might result in a situation where a model performs poorly with new variants or new data, is unstable, and is not applicable, even though it may attain an accuracy which is high with one particular dataset. It is difficult to identify the best AI model and the collection of superior features that can accurately identify complex new malwares. Gaber et al. (2024) said that in order to build an AI model that might be applied in a real-world situation, future research should focus on obtaining genuine characteristics from a top-notch repository [41].

3.2 Model Defination

3.2.1 Random Forest

The Random Forest algorithm is a powerful tree learning tech-nique. It creates several decision trees throughout the training period. By using a random subset of the data set each tree is constructed for analyzing a random subset of attributes for each section. This unpredictability reduces likelihood of overfitting and improves prediction accuracy overall by adding variation among individual trees
This method aggregates each prediction tree's output. This is done by either voting (for classification type tasks) or by taking averages (for regression type tasks). This joint method of making decisions, which benefits from the insights of several trees, offers precise and dependable results. Random forest can handle very complex data, reduce overfitting issues, and generate reliable predictions for a range of circumstances.

3.2.2 Support Vector Machine (SVM)

For both regression and classification problems, a popular supervised learning model is the Support Vector Machine (SVM). Nonetheless, it is mostly applied in machine learning classification problems. The SVM method's objective is to determine the optimal decision boundary for classifying an n-dimensional space so that further data points may be quickly added to the relevant category in the future. The vectors and extremums that help form the hyperplane are chosen via SVM. This technique is called Support Vector Machine, and these extremums are called support vectors. SVMs are of two categories :
Linear SVM: It is used for data that are separable linearly.
Non-linear SVM: It is used for data that are separable but non-linearly.

3.2.3 Logistic Regression

Logistic regression is a supervised machine learning technique for classifying issues with the objective to forecast the probability of an instance belonging to a certain class. Used in situations involving binary classification. Here, the sigmoid function is used for binary classification. This turns independent variables into probability values between 0 and 1. While logistic regression forecasts probability, linear regression predicts continuous val-ues. By setting a threshold, usually 0.5, logistic regression converts the probability output into a class label (either 0 or 1).

3.2.4 K-Nearest Neighbor (KNN)

KNN is an important ML classification technique which falls under supervised learning category. It is substantially used for intrusion detection. Due to being non-parametric, it is frequently employed in real-world applications. Here, we make advantage of training data that groups coordinates according to a characteristic. Classification difficulties are the main use for it. When the existing data and new data are analogous, the KNN method assigns the new data to the category that most closely matches the ones that already exist. The KNN method employs similarity to classify incoming data points while maintaining all of the existing data. This implies that the KNN method may be used to quickly classify newly received data into the relevant category.

3.2.5 Neural Network

Neural network is an ML model which uses layered-arrangement of connected nodes that resemble the organization of human brains. NN creates a flexible structure that allows computers to grow continuously by learning from their errors. These artificial neurons i.e. nodes are basically software modules known as nodes. It generally has three layers:
Input Layer: This layer takes inputs from the outside world and then processes, analyzes and passes the data to the next layer.
Hidden Layer: A neural network may have a very large number of hidden layers where a hidden layer takes input either from the input layer or from its preceding hidden layer if there is any.

Output Layer: This layer gives the final output. For multiclass classificatons, the output layer can have multiple output nodes.

3.2.6 Decision Tree

Decision Tree is a non-parametric supervised learning method for modeling and forecasting results from input data in machine learning. With every internal node evaluating an attribute, every branch signifying an attribute value, and every leaf node indicating the final assessment or forecast, it resembles a hierarchical tree. The supervised learning domain includes the decision tree method. Decision trees can be employed for both classification and regression problems. To find out the optimal split points inside a tree using a greedy search, divide and conquer method is employed by decision trees. After that, this division process is carried out recursively in a top-down approach until all or the records' majority portion are tagged with class labels that are specific.

Chapter 4

Dataset Description

This research was conducted on four individual threats with four different datasets. The datasets and the processing of them are described in this chapter.

4.1 Dataset for DDoS Detection

4.1.1 Data analysis of DDoS dataset

This dataset, which comes from Mendeley Data, is called the DDoS Attack SDN Dataset. The Mininet emulator was used to create this dataset, which is especially intended for Software Defined Networking (SDN). It provides a basis for classifying traffic using deep learning and machine learning methods. Ten distinct Mininet topologies are created at the beginning of the project, each having switches that are connected to a single Ryu controller. In addition to hostile traffic types including TCP SYN assaults, UDP Flood attacks, and ICMP attacks, the network simulation also includes benign TCP, UDP, and ICMP data. There are 23 features in all, with malicious traffic being labelled as 1 and benign traffic as 0. Ultimately, there are 104,345 rows of data stored in this dataset, which is formatted as a .csv file. Figure 4.1 shows that 60.9% are malicious and 39.1% are benign requests.

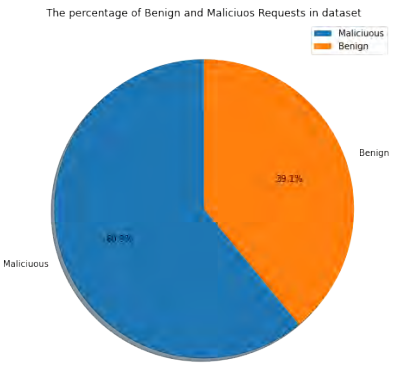

Figure 4.1: The percentage of Benign and Malicious Requests

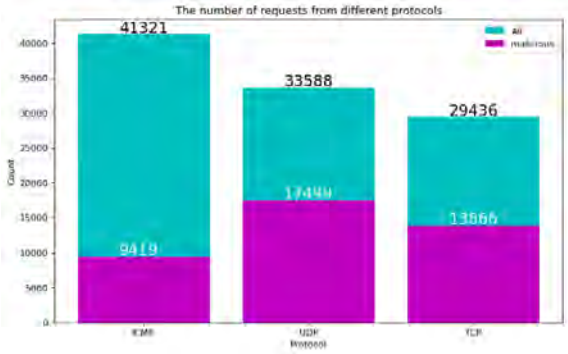

Figure 4.2: The number of requests from different protocols

This figure 4.2 here illustrates a comparison of the total number of requests and the number of malicious requests associated with the ICMP, UDP, and TCP protocols. ICMP shows the highest total request count, totaling 41,321 requests, with 9,419 classified as malicious. This suggests that, despite the high volume of ICMP traffic, the proportion of malicious traffic remains relatively low compared to legitimate usage. UDP follows closely with a total of 33,588 requests, among which 17,499 are identified as malicious. This underscores UDP's status as a common attack vector, likely due to its connectionless protocol. TCP accounts for 29,436 total requests, with 13,866 being malicious. The inherent handshaking mechanism of TCP may result in fewer malicious attempts in contrast to UDP.

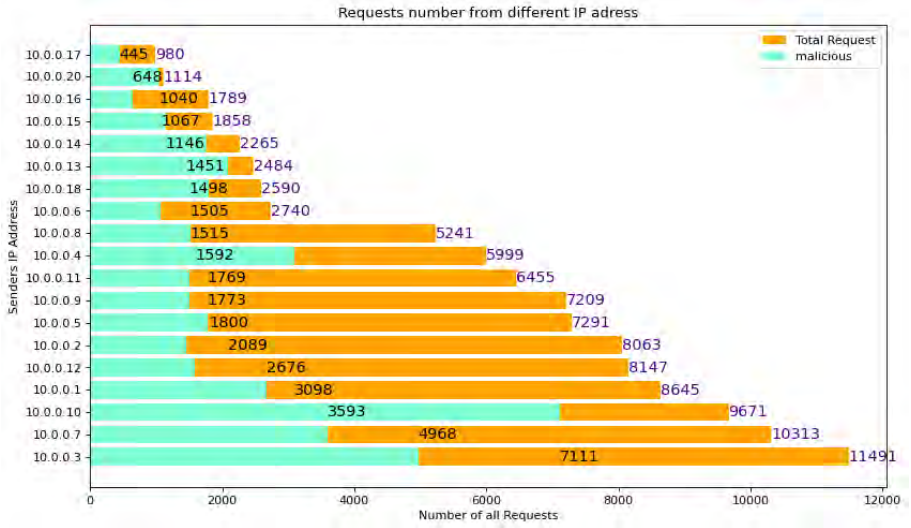

Figure 4.3: Total Number of Requests

This figure 4.3 represents requests from various source IP addresses, distinguishing between malicious and total requests. Certain IP addresses, such as 10.0.0.3 and 10.0.0.7, demonstrate substantial activity with over 10,000 total requests. Furthermore, these IP addresses exhibit a considerable amount of malicious traffic (7,111 and 4,968, respectively), suggesting they may be potential sources of DDoS attacks. In contrast, IP addresses with lower traffic, like 10.0.0.20, exhibit a smaller ratio of malicious requests to total requests, making them less likely to raise suspicion.

Feature	Description	D-type
dt	Timestamp of flow	int64
switch	Switch ID	int64
src	IP address of the source	object
dst	IP address of the destination	object
pktcount	The quantity of packets	int64
bytecount	Number of bytes	int64
dur	Duration of flow in seconds	int64
dur nsec	Duration of flow in nanoseconds	int64
tot dur	Total time spent in flow	float64
flows	Number of flows	int64
packetins	Number of packet-ins	int64
pktperflow	Each flow's packets	int64
byteperflow	Each flow's bytes	int64
pktrate	Rate of packets	int64
Pairflow	Pair flow ID	int64
Protocol	Protocol type	object
port no	Port number	int64
tx bytes	Transmitted bytes	int64
rx bytes	Received bytes	int64
tx kbps	Transmit rate in kilobits per second	int64
rx kbps	Receive rate in kilobits per second	float64
tot kbps	Total rate in kilobits per second	float64
label	Flow label	int64

Table 4.1: Summary of the Features of the Dataset

Table 4.1 shows that dataset comprises 23 features that capture intricate network flow characteristics. Notable attributes include flow timestamps (dt), identifiers (switch, src, dst), and traffic metrics such as packet count, byte count, and flow durations (duration, total duration). Performance metrics encompass packet and byte rates (packet rate, bytes per flow, transmission kbps, reception kbps, total kbps). Features like protocol and port number delineate communication details, while flow statistics and packet input provide insights into flow dynamics

4.1.2 Data Preprocessing of DDoS Attack SDN Dataset

Data Preprocessing is one of the most crucial parts of the experimentation and model implementation. The DDoS Attack SDN Dataset consists of both numerical and categorical data Since the dataset is in .csv format it's easier to work. We have utilized multiple techniques for a better understanding of the dataset including explainable AI and the most effective way to process the data.

Label Encoding

Label encoding is applied to the categorical variable Protocol. Label encoding is a preprocessing step where unique string values in a categorical column are converted into numeric representations. Specifically, the Protocol column is treated as a categorical data type, and each unique protocol name (e.g., ICMP, UDP, TCP) is assigned a distinct numeric value. This transformation ensures that the categorical data can be utilized by machine learning algorithms, which typically require numerical inputs. In python, from sklearn we imported label encoder which takes the class label and it replaces it with an assigned numerical value. The encoding is done directly on the dataset, replacing the original text-based protocol names with their corresponding numeric codes.

Identifying and removing irrelevant features using Matrix correlation

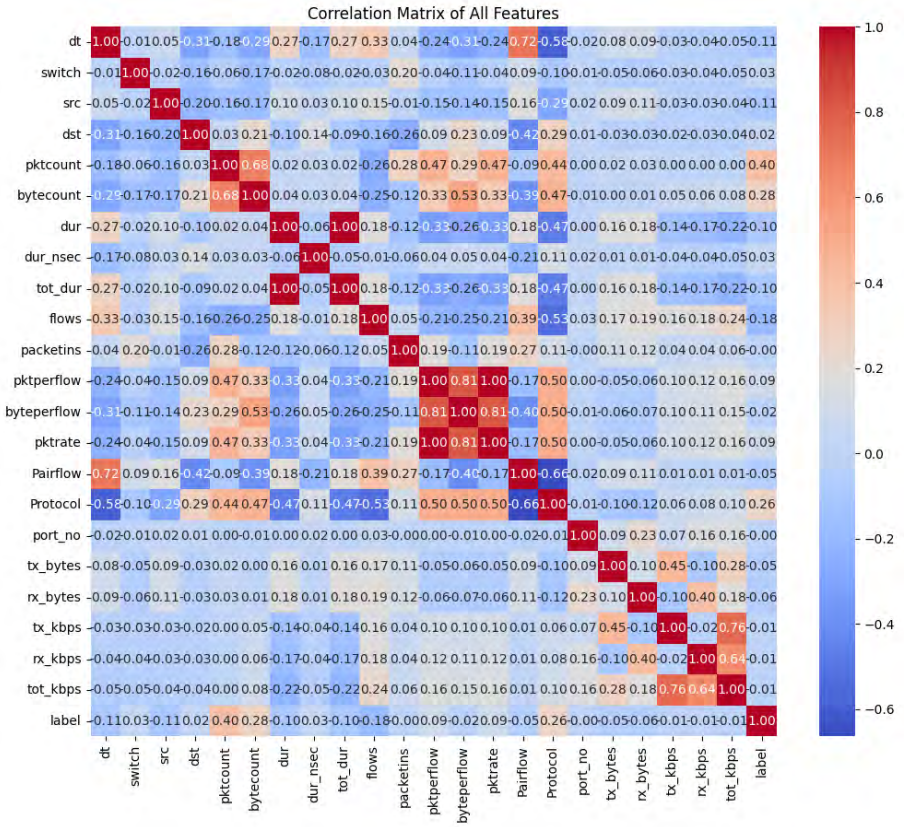

Figure 4.4: Correlation Matrix of All Features

Figure 4.4 is a heatmap, illustrates the correlation between features in your dataset. Features like bytecount and pktcount have a strong correlation (near 0.72). Some features, like switch, src, and dst, exhibit very low correlation with the label (target variable), making them candidates for removal during preprocessing. Features like txbytes and totkbps don't show a moderate correlation with the label, suggesting their non-relevance to detecting DDoS traffic. Irrelevant or highly correlated features can be removed to simplify the model and reduce overfitting. For example: Drop features with a low correlation to the label (0.1). Thus, we dropped txkbps and rsKbps.

Data Balancing

The dataset was not in a balanced state. Thus, we applied SMOTE to balance our dataset. We implemented a python library 'imbalanced-learn'. Class imbalance in the target variable 'label' is addressed by the Synthetic Minority Oversampling Technique (SMOTE). This technique creates synthetic instances for the minority class, thereby ensuring a balanced dataset for training purposes. This crucial balancing step precedes feature selection, as class imbalance can distort the model's evaluation of feature significance. Figure 4.5 shows the balanced distribution of features.

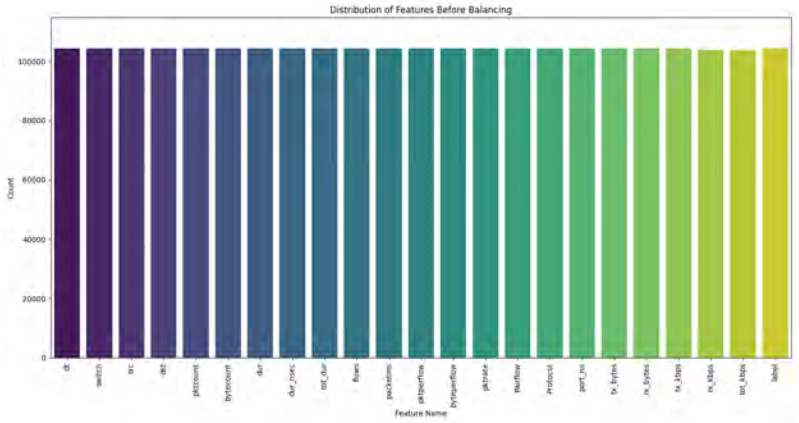

Figure 4.5: Balanced Distribution of Features

Feature Selection

The ExtraTreesClassifier is employed to assess feature importance. By measuring the decrease in impurity (like Gini or entropy) owing to each feature, this ensemble technique uses decision trees to calculate significance scores. The significance of each feature is evaluated based on its capability to differentiate data points within each tree. Importance is determined as the weighted average reduction in impurity across all splits involving the feature.

$$\text{Feature Importance} = \frac{\sum(\text{Reduction in Impurity for a Feature})}{\text{Total Reduction in Impurity}} \tag{4.1}$$

Equation 4.1 represents the process of how the feature importance was determined. The classifier is trained on a balanced dataset (X balanced, y balanced) to facilitate an equitable assessment of feature importance across all classes. Each feature is given an importance score by the model after the training phase, which is determined by the frequency and significance of the feature in data splits inside the decision trees. Following the computation of feature importance scores, we ranked the top 10 features by ordering the importance scores in descending order.

Figure 4.6 shows the top 10 features are: 'byteperflow', 'pktperflow', 'pktrate', 'bytecount', 'pktcount', 'Protocol', 'packetins', 'dur', 'tot dur', 'dt'

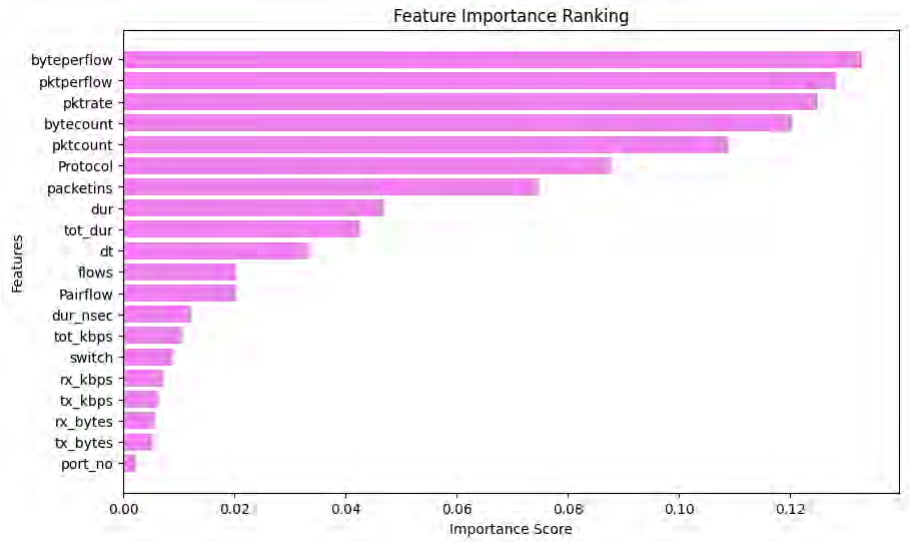

Figure 4.6: Feature Importance Ranking

By inspecting the feature importance ranking it is obvious that the top 10 features are significantly dominant than the other features. Meanwhile, the 11th ranked feature scores less than 0.25 on important metrics which is very low. Thus, those features were not selected.

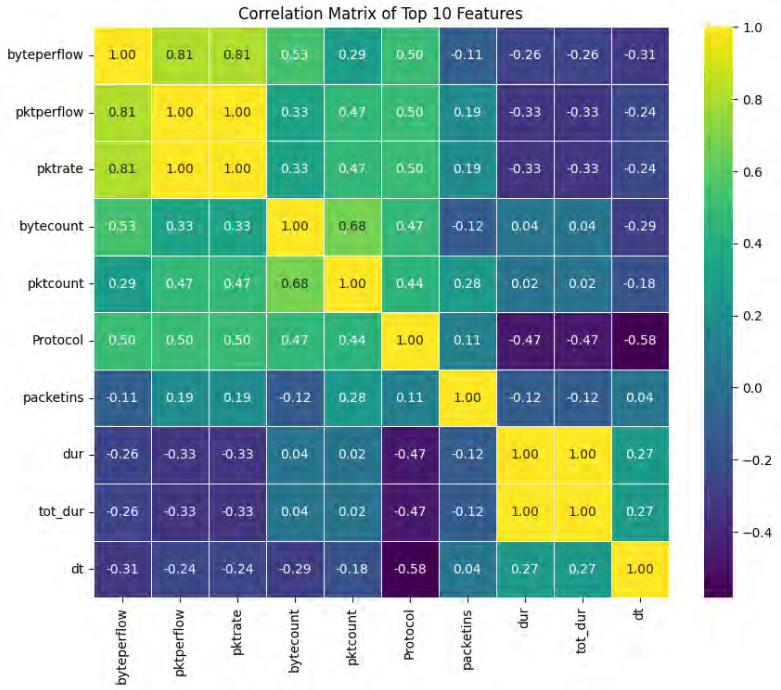

Figure 4.7: Correlation Matrix of Top 10 Features

By inspecting the figure 4.7 the correlation matrix, there are Strongly correlated attributes such as byteperflow, pktperflow, and pktrate encapsulate critical traffic characteristics, while distinct elements like Protocol contribute unique insights regarding communication types. Volume metrics (bytecount, pktcount) and temporal

indicators (tot dur, dt) elucidate data size, frequency, and time-related patterns. This amalgamation of flow, protocol, and temporal metrics guarantees that the model accurately differentiates between standard and anomalous traffic, thereby enhancing classification precision.

Scaling

We have implemented standardscaler in order to scale our dataset. Data is transformed to have a unit variance and zero mean. Therefore, it forbids features with greater scales from influencing the learning process and it improves the models performance.

Data split

The data was divided into two groups. Thirty percent of the data was used to test the model, and seventy percent was used to train it. It was done by train test split from the sklearn library.

4.2 Dataset for Data Breach

4.2.1 Data Analysis

The dataset we used was CSE-CIC-IDS 2018. The Canadian Institute for Cybersecurity (CIC) established it to aid in the advancement of cybersecurity research and development. CSE-CIC-IDS2018 is an extensive Intrusion Detection System (IDS) dataset. In order to replicate real life network activity, the dataset contains both malicious and benign network traffic.

The dataset is ideal for machine learning-based intrusion detection system development and testing because it includes comprehensive network flow data, extracted features, and packet capture (PCAP) files. Additionally, it contains labelled data, which makes it possible to efficiently train and validate models. CICFlowMeter-V3 was used for feature extraction and it outputted the extracted features in CSV format. The CSV format makes it easier for machine learning to analyze.

Table 4.2 represents the 80 features in the dataset. The table shows the features and their descriptions.

Table 4.2: Description of the Features in Data Breach Dataset

Feature	**Description**
bw iat tot	Total time spent in the backward division between two packets
bw iat avg	The interval of time between two packets delivered in reverse
bw iat std	The standard deviation of the time difference between two packets delivered backwards
bw iat max	The maximum amount of time that passes between two packets transmitted backward
bw iat min	The shortest amount of time between two packets transmitted backward
fl dur	Duration of flow
tot fw pk	The total number of packets moving forward
tot bw pk	Total number of packets in the reverse direction
tot l fw pkt	The entire packet size in the forward direction
fw pkt l max	Maximum packet size when moving forward
fw pkt l min	The packet's minimum size when moving forward
fw pkt l avg	The average packet size moving forward
fw pkt l std	The packet's standard deviation size in the forward direction
fw iat tot	The total amount of time that passes between two packets pushed forward
fw iat avg	The interval of time between two packets transmitted forward
fw iat std	The standard deviation of the time interval between two forward-directed packets

Continued on next page

Feature	Description
fw iat max	The maximum amount of time that can pass between two packets sent forward
fw iat min	Minimum interval between two forward-directed packets
fw psh flag	The number of times the PSH flag was set in forward-moving packets (0 for UDP)
bw psh flag	The frequency with which the PSH flag was set in packets moving backward (0 for UDP)
fw urg flag	The frequency with which the URG flag was set in forward-moving packets (0 for UDP)
bw urg flag	The frequency with which the URG flag was set in packets moving backward (0 for UDP)
fw hdr len	Total bytes utilised in the forward direction for headers
bw hdr len	Total bytes utilised in the forward direction for headers
fw pkt s	The quantity of packets forwarded each second
bw pkt s	Backward packet count per second
Bw pkt l max	Maximum packet size when moving backwards
Bw pkt l min	Minimum packet size when moving backwards
Bw pkt l avg	Average packet size when moving backwards
Bw pkt l std	The packet's standard deviation size in reverse
fl byt s	The number of packets transmitted per second is known as the flow byte rate.
fl pkt s	The number of packets transported per second is known as the flow packet rate.
fl iat avg	Average amount of time between two flows
fl iat std	Time two flows' standard deviation
fl iat max	Maximum interval between two flows
fl iat min	The shortest interval between two flows
pkt len min	Minimum flow length
pkt len max	Maximum flow length
pkt len avg	Average flow length
pkt len std	The flow's standard deviation length
pkt len va	Minimum packet inter-arrival time
fin cnt	The quantity of packets that have FIN
syn cnt	Number of SYN-containing packets
rst cnt	Number of RST-enabled packets
pst cnt	Number of PUSH-enabled packets
ack cnt	Number of ACK-containing packets
urg cnt	Number of URG-containing packets
cwe cnt	Number of CWE-containing packets
ece cnt	Number of ECE-containing packets
down up ratio	ratio of downloads to uploads
pkt size avg	Average packet size
atv avg	Before going idle, the average time the flow was active
atv std	The standard deviation of the duration of an active flow before it became idle
atv max	The amount of time a flow was active before ceasing to exist

Continued on next page

Feature	Description
atv min	The amount of time a flow was operational before it stopped
idl avg	A flow was inactive in the interim before becoming active.
idl std	Standard deviation of the interval between an idle and active flow
idl max	The longest period of inactivity before a flow became active
idl min	The amount of time a flow was inactive before activating
fw seg avg	Average packet size seen moving forward
bw seg avg	Average size seen when moving backward
fw byt blk avg	The average number of bulk bytes moving forward
fw pkt blk avg	The average bulk rate of packets in the forward direction
fw blk rate avg	The average number of bulk rates moving forward
bw byt blk avg	The average bulk rate of bytes in the backward direction
bw pkt blk avg	Average bulk rate of packets in the backward division
bw blk rate avg	The average bulk rate in the reverse direction
subfl fw pk	The mean quantity of packets in a forward-directed subflow
subfl fw byt	The mean quantity of bytes in a forward-moving subflow
subfl bw pkt	The mean quantity of packets in a subflow moving backward
subfl bw byt	The mean quantity of bytes in a backward-flowing subflow
fw win byt	The number of bytes transmitted in the forward division's first window
bw win byt	The quantity of bytes transmitted in the backward direction during the first window
Fw act pkt	Number of forward-bound packets containing at least one byte of TCP data payload
fw seg min	The smallest segment size seen moving forward.
dst port	Destination port number of a network flow

4.2.2 Data Preprocessing

Data Balancing

The dataset had 1,048,575 samples where 762,384 were benign and 286,191 were attacks. In other words, 72.71 percent were benign and 27.29 percent were attacks. The dataset was very large and imbalanced which can cause overfitting and take a lot of computation time. Therefore, we used the "resample" utility function from the "sklearns.utils" module to resample and balance the dataset, taking 50,000 samples total, with 25,000 samples each class. Figure 4.8 shows the state of datasets before it was balanced and figure 4.9 shows that after balancing both attack and normal requests are equally 50%.

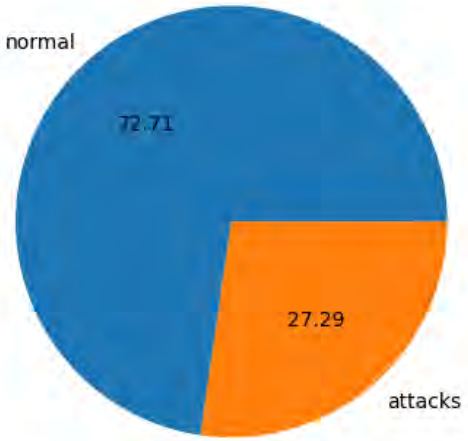

Figure 4.8: Dataset before balancing

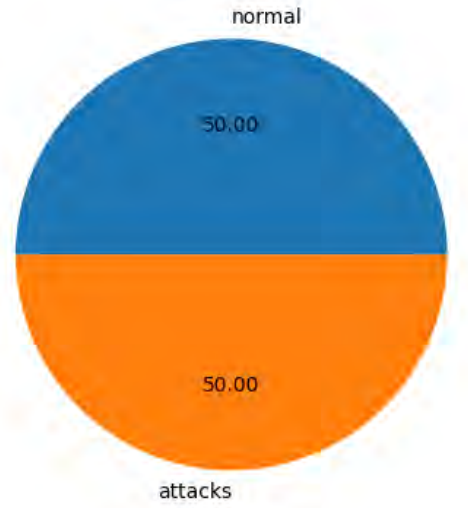

Figure 4.9: Dataset after balancing

Feature Selection

The dataset had 80 features. We selected the best 10 features using RFE where a Random Forest model was used as the base estimator. The best 10 features were Dst Port', 'Flow Duration', 'Bwd Pkt Len Mean', 'Flow Pkts/s', 'Flow IAT Mean', 'Fwd IAT Tot', 'Fwd IAT Mean', 'Fwd IAT Max', 'Bwd Seg Size Avg' and 'Init Fwd Win Byts'. Recursive Feature Elimination, or RFE for short, is a feature selection method that chooses the most important features to enhance the effectiveness and performance of machine learning models. To find the most predictive subset, the least significant features are iteratively eliminated, and a model is then constructed using the remaining features. Model performance, training speed, and interpretability can all be enhanced by feature reduction.

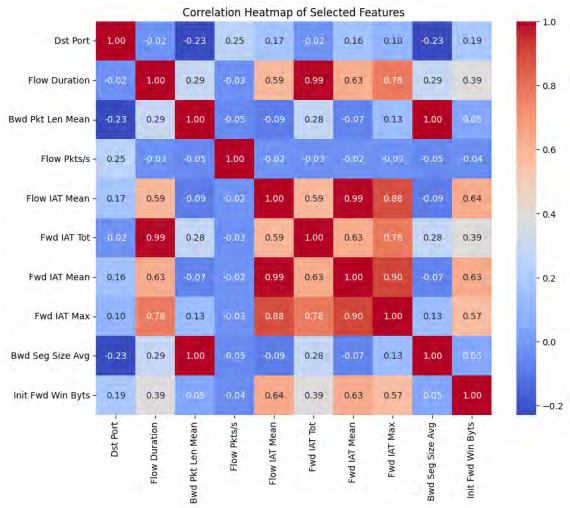

Figure 4.10: Correlation Matrix of Best 10 Features in Data Breach Dataset

Figure 4.10 is the correlation matrix where we see that "Flow Duration" has strong positive correlations with "Fwd IAT Tot," "Fwd IAT Mean," and "Fwd IAT Max." This implies that these features are closely related with the flow's total duration. "Flow IAT Mean" has strong positive correlations with "Fwd IAT Tot," "Fwd IAT Mean," and "Fwd IAT Max." This suggests that the maximum, minimum, and total inter-arrival periods are all strongly correlated with the average time between packets within a flow. "Bwd Pkt Len Mean" has a strong negative correlation with "Dst Port." This implies that the destination port number and the average length of backward packets are inversely related.

Scaling

To make sure that every feature is on the same scale, which aids in the model's efficient learning, we employed a standard scaler. Scikit-learn's StandardScaler preprocessing tool standardizes features by scaling them to unit variance and eliminating the mean.

Data Split

We have split the dataset using the "train_test_split" tool from the "sklearn" library where 20 percent is for testing and 80 percent is for training. A decent balance is achieved by allocating 20 percent for testing and 80 percent for training. It ensures the model has sufficient data for learning. And that the testing set is large enough to provide reliable performance metrics. Reserving 20 percent for testing, strikes a balance where the training set is large enough to reduce overfitting.

4.3 Dataset for Malware

4.3.1 Dataset Description

The dataset we used had 138,047 samples, including 41,323 legitimate normal binary (exe,dll) files and 96,724 malware files obtained from virusshare.com. Therefore 70.1 percent are malicious and 29.9 data are benign. Figure 4.11 shows the percentage of malicious and benign data and it also represents the dataset is not balanced.

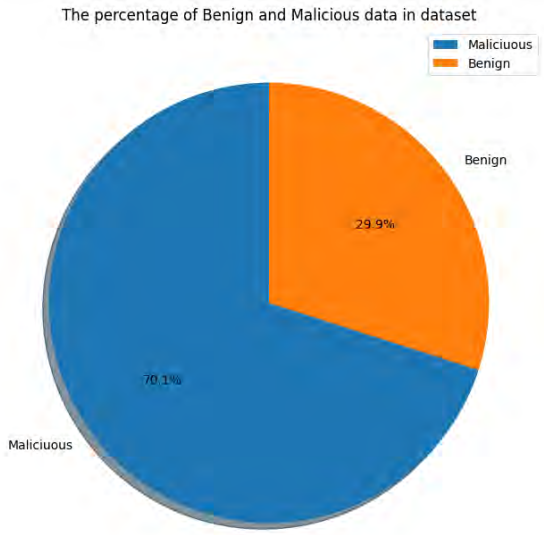

Figure 4.11: The proportion of malicious and benign data in the malware dataset

The dataset is structured as a CSV file containing metadata about files related to malware and legitimate software, which are useful for machine learning applications in penetration testing. The dataset contains around 55 features that provide various characteristics of the files, such as their headers, sections, imports, and metadata used for analysis. The table 4.3 below shows the important features and their description:

Table 4.3: Description of the Features in Malware Dataset

Feature	**Description**
SizeOfUninitializedData	Byte count of the uninitialised data segment.
AddressOfEntryPoint	The file's entry point address.
BaseOfCode	The code section's base address.
BaseOfData	The data section's base address.
ImageBase	preferred location for the image's first byte when it is loaded into memory.
SectionAlignment	Section alignment upon memory loading.
FileAlignment	alignment of the file's raw data.
MajorOperatingSystemVersion	A major operating system version is necessary.
MinorOperatingSystemVersion	A minimal operating system version is necessary.
MajorImageVersion	The image's major version number.
MinorImageVersion	The image's minor version number.
MajorSubsystemVersion	version of the subsystem that is major.
MinorSubsystemVersion	A subsystem version that is smaller.
Machine	type of machine that the file is intended for.
SizeOfOptionalHeader	size of the file's optional header.
Characteristics	file attributes, including executable flags.
MajorLinkerVersion	The linker's major version number.
MinorLinkerVersion	The linker's minor version number was used.
SizeOfCode	The code section's size in bytes.
SizeOfInitializedData	Byte size of the initialised data segment.
SizeOfImage	whole image size when it is loaded into memory.
SizeOfHeaders	total size of the file's headers.
CheckSum	image checksum.
SectionsMinEntropy	Sections' minimum entropy.
SectionsMaxEntropy	maximum sectional entropy.
SectionsMeanRawsize	Average raw data size by section.
SectionsMinRawsize	Sections' minimum raw data size.
SectionMaxRawsize	Sectional raw data maximum size.
SectionsMeanVirtualsize	Average virtual portion size.
SectionsMinVirtualsize	minimum sectional virtual size.
SectionMaxVirtualsize	Sections' maximum virtual size.
ImportsNbDLL	quantity of DLLs imported.
ImportsNb	total quantity of imports.
ImportsNbOrdinal	ordinal import count.
ExportNb	quantity of functions that are exported.
ResourcesNb	The file's resource count.
ResourcesMeanEntropy	mean resource entropy.
ResourcesMinEntropy	minimum resource entropy.
ResourcesMaxEntropy	maximum resource entropy.
ResourcesMeanSize	average resource size.
Subsystem	Execution-related subsystem (e.g., GUI, console).
DllCharacteristics	DLL file characteristics.
SizeOfStackReserve	Bytes of the stack size to be reserved.

Continued on next page

Feature	Description
SizeOfStackCommit	Bytes of the stack size to commit.
SizeOfHeapReserve	Heap size to be reserved in bytes.
SizeOfHeapCommit	Heap size in bytes to commit.
LoaderFlags	loader flags for optimisation and debugging.
NumberOfRvaAndSizes	The optional header's number of directory items.
SectionsNb	number of the file's sections.
SectionsMeanEntropy	All sections' mean entropy (used to identify packed code).
ResourcesMinSize	minimum resource size.
ResourcesMaxSize	maximum resource size.
LoadConfigurationSize	The load configuration's size.
VersionInformationSize	Version information resource size.
legitimate	A binary label that indicates if the file is malicious (0) or not (1).

4.3.2 Data Preprocessing of Malware Dataset

Data Cleaning

We have removed the columns 'Name' and 'md5' from the dataset. The Name and md5 columns are irrelevant features. They do not provide information that helps the model making predictions. Including them might lead to overfitting, as the model could learn spurious relationships based on these identifiers rather than generalizable patterns.

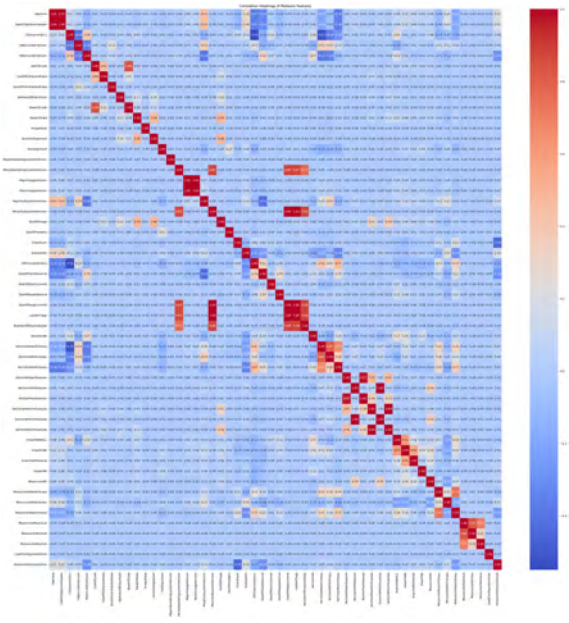

Figure 4.12: Correlation Matrix of All Features in Malware Dataset

Figure 4.12 is the matrix correlation where the SectionMaxEntropy and SectionsMeanSize , NumberOfSections and ResourcesMinEntropy have strong negative correlation. Significant trade-offs or patterns in malware characteristics can be revealed by features with strong negative correlations. SectionEntropy and SectionMinEntropy, ImportsNbOrdinal and ImportsNb, SectionsMeanEntropy and SectionsMax-

Entropy have strong positive correlation. Redundancy may be indicated by features that have a positive correlation.

Feature Selection

We used Principal Component Analysis (PCA) with 95 percent variance for feature selection. Principal components are an entirely new set of uncorrelated features that are created by PCA from the original features. The original features are combined linearly to create these components. 'PCA' tool was imported from 'sklearn.decomposition' library.

Scaling

Standard Scaler was used to ensure that every feature is on the same scale. 'StandardScaler' tool was imported from 'sklearn.preprocessing' library.

Data Split

The "train_test_split" tool from the "sklearn" library was used to divide the dataset into 80 percent for training and 20 percent for testing.

4.4 Dataset for Phishing Attack

4.4.1 Data Analysis

A critical step in training and assessing any machine learning model is dividing the data into subsets for testing and training. The **test set** is set aside to evaluate the model's performance on unknown data, whereas the **training set** is used to train the model. Instead of just memorising patterns in the training set, this method makes that the model generalises well to new, unknown data, preventing **overfitting**.

Either **70:30** or **80:20** are popular split ratios for data division, with the larger portion usually being used for training and the smaller portion for testing. This split is made possible by the `train_test_split` function from the `sklearn.model_selection` module. In order to train the machine learning model and assess its performance, this function splits the data set into two subsets.

Using this method ensures that the model's accuracy and reliability are measured independently of the data it was trained on, providing a fair assessment of its performance.

Table 4.4 displays the relevant features that are needed to detect phishing. The dataset comprises 30 features designed to analyze website characteristics and detect phishing. These features assess various aspects of a website's structure, behavior, and domain properties. For example, **URL_Length** identifies suspiciously long URLs often used in phishing attempts, and **Shortining_Service** checks whether a URL shortening service, such as `bit.ly`, is utilized to hide the original domain. Features like **SSLfinal_State** calculate the validity of SSL certificates, which are important for establishing the credibility of a website.

Feature	Description	Importance
URL_Length	Checks if URL length is suspicious.	High
Shortining_Service	Indicates if a URL shortening service was used.	Medium
having_At_Symbol	Checks for "@" in the URL.	High
SSLfinal_State	Assesses SSL certificate legitimacy.	High
Domain_registeration_length	Evaluates domain registration duration.	Medium
Favicon	Checks if favicon is from an external domain.	Low
Request_URL	Checks if resources are from the same domain.	High
URL_of_Anchor	Measures external links in anchor tags.	Medium
on_mouseover	Detects behavior changes on mouse hover.	Medium
Iframe	Checks for invisible iFrames.	High
DNSRecord	Verifies DNS record validity.	High
Google_Index	Checks if the website is indexed by Google.	High
Result	Target variable: phishing (1) or legitimate (-1).	-

Table 4.4: Summary of the Features for Phishing Detection

These domain-related features provide insights into **Domain_registeration_length** and **DNSRecord**, which are indicators of domain credibility. Short registration durations or invalid DNS records are commonly associated with phishing attempts.

Behavioral features like **Request_URL** and **Iframe** further aid phishing detection:

- **Request_URL** checks if external resources are loaded from different domains.
- **Iframe** detects the presence of invisible iFrames, frequently used in phishing schemes to mislead users.

Additionally, **Google_Index** verifies whether a website is indexed by Google. Legitimate websites are much more likely to appear in search results, while phishing websites tend to be unindexed.

The target variable, Result, is binary, where 1 represents phishing websites and -1 represents legitimate ones. Most of the features are numeric and related to the detection of patterns that identify phishing from legitimate websites. This dataset offers a thorough basis for developing machine learning models that can efficiently identify phishing attempts.

4.4.2 Data Preprocessing of Phishing Attack CSV Dataset

The pre-processing of data is an important step in the preparation of the Phishing Attack CSV Dataset for machine learning. Libraries are first imported, followed by the loading of the dataset into a pandas DataFrame. Checking for missing values is the next step, with either the rows containing NaN being dropped or the values filled. The categorical variables such as the target variable 'Result' are encoded to

numerical values using LabelEncoder. Afterwards, numerical features are standardized using StandardScaler for uniform scaling that makes convergence better for the model.
To successfully validate model performance, the data is then divided into training and testing subsets using `train_test_split`. Optionally, low-variance features that may contribute little to the predictive power of the model are removed using `VarianceThreshold`. Finally, the preprocessed data is checked to confirm if it is clean, scaled, and ready for machine learning model training. These preprocessing steps ensure that the data is optimally structured for accurate and efficient model training and evaluation.

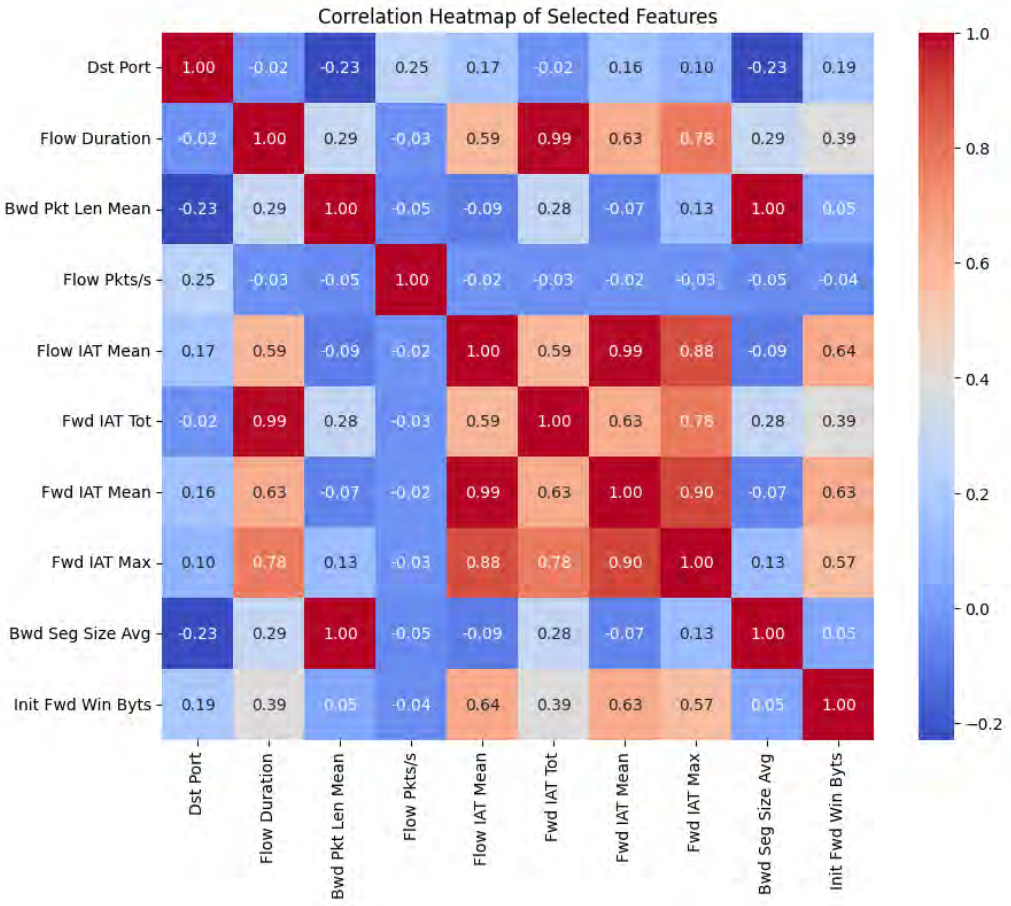

Figure 4.13: Heatmap with the Mask and Correct Aspect Ratio

Figure 4.13 is a correlation matrix heatmap, a graphical representation of data employed in the visualization of the correlations among variables. In the context of the phishing dataset, the heatmap shows how strongly each feature is related to other features and the target variable, `Result`. The intensity of the colors corresponds to the correlation coefficient, ranging from -1 to 1.

Shades of blue indicate a positive correlation, which indicates that when one variable rises, the other tends to rise as well. On the other hand, a negative correlation, which is represented by red hues, would indicate that one factor tends to decrease as the other increases. No association is indicated by a value near 0.

Heatmaps provide insight into feature relationships and their relationship with the target variable. In this case, `SSLfinal_State`, `having_Sub_Domain`, and `URL_Length` are some features likely to present a strong correlation with phishing behavior. Additionally, heatmaps can show redundancies; features highly correlated with one another may contribute less to information in machine learning models.

By visualizing the pattern of features, heatmaps help prioritize the most informative ones while also assessing probable relationships that could impact model performance.

Label Encoding
The process of transforming categorical variables into numerical values so that machine learning models can comprehend them is known as label encoding. In the context of the phishing dataset, the target variable Result contains two classes: 1 phishing and -1 for legitimate websites. Label Encoding simply assigns these unique numerical values to represent the categories. Examples could be 1 for phishing and -1 for legitimate. This transformation makes sure that most machine learning algorithms can process and be trained on the target variable effectively, since most algorithms cannot work with non-numeric data directly.

Data Balancing
Imbalanced data is a common problem that frequently arises in many classification tasks, including phishing detection. Typically, one class may dominate the other; for instance, there may be more phishing web pages than legitimate ones. For a model to learn from both classes well and not bias toward one, balancing of data should be considered. Techniques of balancing can be done through oversampling, undersampling, or SMOTE. SMOTE generates synthetic data points of the minority class by creating plausible variations that present the model with equal examples of each class. This, in turn, enhances its generalizing ability and provides better predictive ability on both classes.

Feature Selection
Finding the most critical features while eliminating superfluous or irrelevant ones is known as feature selection. It enhances model performance by reducing training time, minimizing overfitting, and focusing only on significant predictors. Common methods of feature selection include low-variance feature removal, correlation-based selection, statistical tests, and Recursive Feature Elimination (RFE). Besides, tree-based algorithms like Random Forest also provide feature ranking. For example, low-variance features, which barely change across observations, are unlikely to bear any predictive value and can be safely removed using VarianceThreshold. By doing this, only significant features will be used to train the models, improving their accuracy and interpretability.

Data Split
A crucial stage in the creation of any machine learning model is the division of the data into training and testing subsets. The test set is set aside to gauge the model's performance on unobserved data, while the model is trained on the training set. By avoiding overfitting, this procedure makes it possible for the model to perform

effectively when applied to new data.

70:30 is used for the split ratios for data division, with the larger portion usually being utilised for training and the smaller portion for testing. For this, the `train_test_split` function from the `sklearn.model_selection` module is frequently utilised. In order to train the machine learning model and evaluate its performance, it splits the dataset into two subgroups. This guarantees that the accuracy and dependability of the model are assessed separately from the data used for training.

Chapter 5

Model Implementation and Results Analysis

5.1 Results

5.1.1 DDoS Detection Results

Result Metrics

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.9994	0.9998	0.9997	0.9998
KNN	0.9701	0.9695	0.9703	0.9699
Logistic Regression	0.7384	0.7158	0.7853	0.7490
TPOT Model	0.9850	0.9723	0.9982	0.9851
Optimized TPOT Model	0.9999	0.9999	0.9999	0.9999

Table 5.1: Models performance based on Top 10 features

A thorough comparison of five machine learning models assessed using the four performance metrics of accuracy, precision, recall, and F1 score is shown in table 5.1. Among the models assessed, the Proposed Optimized TPOT Model consistently surpasses all others by attaining near-perfect scores across all metrics. The Random Forest model ranks as the second-best performer, and LR displays the lowest performance. This table emphasizes the advantage of automated optimization (TPOT) compared to traditional models, highlighting its potential for enhancing predictive performance. This evolutionary algorithms parameters were optimized, the generation parameter was 5 and the population was 50 whereas the cross-validation was 5, which maintained a proper balance, and the config dictionary was "tpot light" so that it works faster than other evolutionary algorithms, and the verbosity was 2.

Metric Comparison:

Accuracy: The Optimized TPOT Model (0.9999) surpasses Random Forest (0.9994), highlighting its superior prediction accuracy.

Precision: The Optimized TPOT attains 0.9999, marginally exceeding Random Forest's 0.9998 in correctly identifying positive instances.

Recall: Boasting a score of 0.9999, the Optimized TPOT outperforms Random Forest (0.9997), demonstrating better sensitivity.

F1 Score: The Optimized TPOT Model (0.9999) exceeds Random Forest's 0.9998, effectively balancing precision and recall.

Learning Curve of Optimized TPOT Model

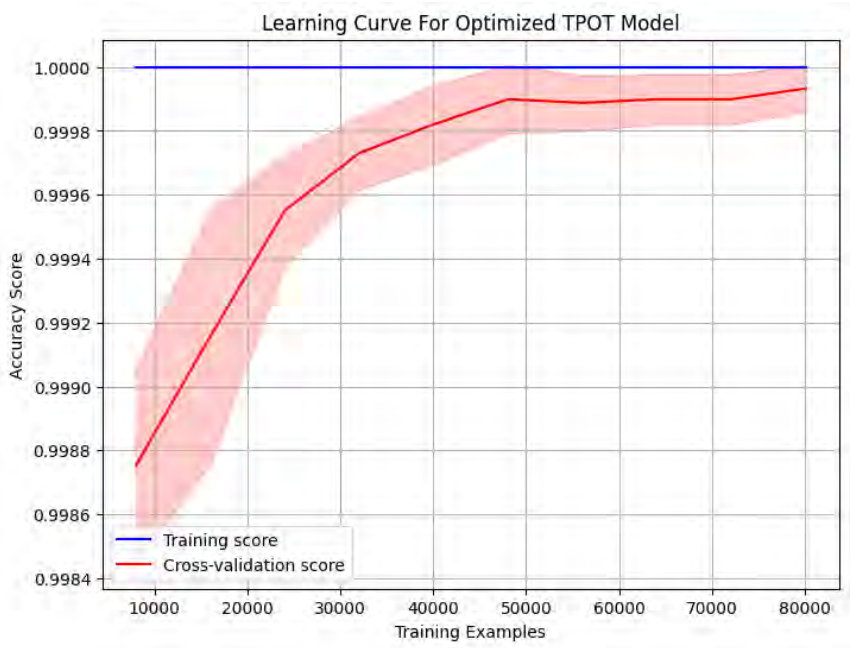

Figure 5.1: Learning Curve of Optimized TPOT Model

Figure 5.1 is the learning curve that illustrates the performance of an optimized TPOT model. The blue line represents the training score, which remains consistently perfect (1.0) across all training sizes, indicating no underfitting. The cross-validation score, represented by the red line, steadily improves as the quantity of training instances rises. The shaded red region highlights the variability in cross-validation scores due to different validation folds. The narrowing of this region as training size grows reflects reduced variance, suggesting better generalization. Overall, the model demonstrates excellent accuracy and stability, with high alignment between training and cross-validation performance at medium dataset sizes.

Selected Features impact on Model (SHAP AI)

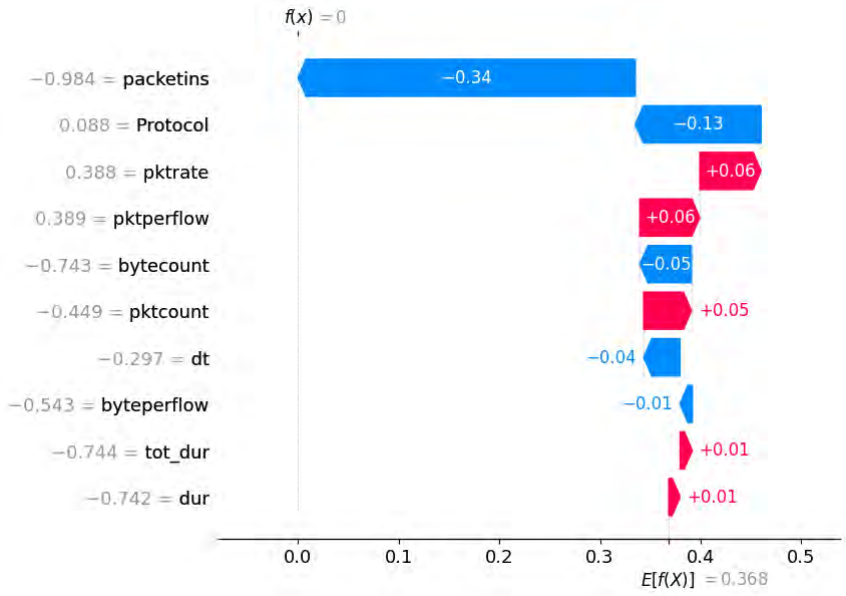

Figure 5.2: Selected Features Effect (SHAP AI)

In figure 5.2 the SHAP waterfall plot illustrates the contributions of the top 10 features identified through feature ranking for DDoS detection, utilizing optimized TPOT model. The features are organized by their significance, demonstrating how each one influences the model's predictions. The features represented by blue bars (e.g., packetins and bytecount) exert a negative impact on the prediction, thereby decreasing its value, whereas those indicated by red bars (e.g., Protocol and pktrate) offer a positive effect, elevating it.

The feature with the highest impact, packetins, exhibits a substantial negative contribution of -0.34, underscoring its essential role in detecting potential DDoS traffic. On the other hand, the highest positive contribution is from the feature pktrate which is 0.06. Other features, such as pktperflow and pktcount, provide smaller but meaningful contributions that further enhance the model's output. The cumulative effect of all these features culminates in the final prediction value, offering a comprehensible perspective on the model's decision-making process.

Comparative analysis

Serial No.	Research Study	Accuracy
1	Ko et al., 2020 [18]	95.24%
2	Han et al., 2018 [9]	96%
3	Myint Oo et al., 2019 [14]	97%
4	Auhoja, 2021 [21]	98.8%
5	Perez-Díaz et al., 2020 [19]	95%
6	Proposed Optimized TPOT Model	**99.99%**

Table 5.2: Comparison of Accuracy Across Studies

This table 5.2 here shows a comparative analysis of testing accuracies from various research studies showing the accuracy of detecting DDoS attacks on similar kinds of datasets. These studies are all based on static environments. However, Ahuoja (2021) achieved a noteworthy accuracy of 98.8%, indicating consistent advancements in this field. Significantly, the Proposed Optimized TPOT Model exceeded all prior studies, attaining an extraordinary accuracy rate of 99.99%. The TPOT model's near-perfect accuracy highlights its applicability in real-world scenarios that demand both precision and reliability. By automating processes such as feature selection, parameter tuning, and model optimization, TPOT outperforms conventional techniques, representing a notable advancement in research related to DDoS attack detections.

5.1.2 Data Breach Results

Result Metrics

Metrics	RF	SVM	LR	NN	Optimized SVM
Accuracy	0.9995	0.7351	0.9690	0.9981	0.9946
Precision	0.9995	0.8266	0.9696	0.9981	0.9946
Recall	0.9995	0.7351	0.9690	0.9981	0.9946
F1 Score	0.9995	0.7157	0.9690	0.9981	0.9946

Table 5.3: Performance metrics for different models

Table 5.3 represents Random forest has the highest accuracy 99.95 percent and SVM has the lowest accuracy 73.51 percent. Random Forest has the highest precision and recall with 99.95 percent for both, indicating that it makes fewer false positive and false negative predictions compared to the other models. Logistic Regression and Neural Network also have high precision and recall. SVM has relatively lower precision 82.66 percent and recall 73.51 percent compared to the other models. An implementation of hyperparameter tuning can improve the SVM model's performance by a large margin.

The findings show that the model with hyperparameter tuning consistently performs better than the current model across all metrics. To be more precise:

Accuracy: The adjusted model reaches a top accuracy of 0.9946, a significant enhancement compared to the current model's 0.7351, highlighting its overall capability to accurately categorize instances.

Precision: The adjusted model's performance of 0.9946 surpasses the current model's 0.8266 by a notable margin. This suggests that the modified model reduces false positives and detects true positive cases more effectively.

Recall: The adjusted model also performs well in recall, achieving a score of 0.9946 compared to the 0.7351 of the current model, showing enhanced ability to identify real positive cases.

F1-Score: The modified model outperforms the present model with an F1-Score of 0.9946, indicating that precision and recall are balanced. This underscores its superior classification performance in general.

The importance of hyper-parameter tuning in optimizing the SVM model's performance is highlighted by the notable enhancements in all metrics. By tweaking factors like kernel type, regularization strength (C), and gamma, the optimized model shows improved resilience and ability to predict accurately. This comparison

illustrates the impact of a systematic tuning process on turning a basic model into a powerful classifier.

Learning Curve

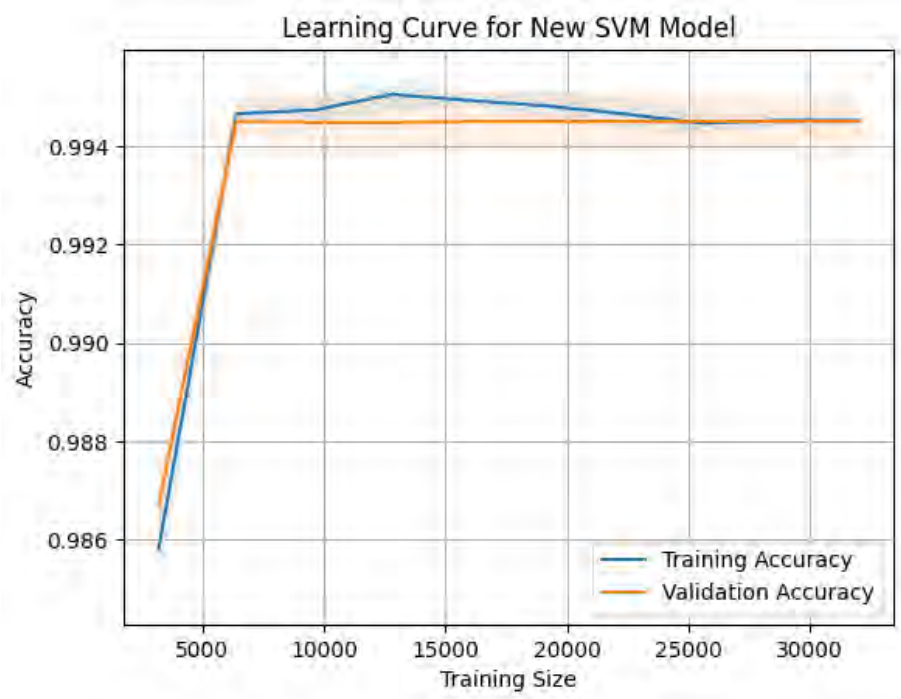

Figure 5.3: Learning Curve of Improved SVM model

From the figure 5.3 which represents the learning curve, we can conclude that:

Low Variance: Across the training size range, there is a consistent and comparatively small difference between the training and validation accuracy. This suggests that the model is successfully generalising to new data and is not overfitting.

Convergence: The curves appear to have levelled off, indicating that the model's performance may not be considerably increased by additional training data.

High Accuracy: The model is clearly learning a significant pattern from the data, as evidenced by the high training and validation accuracy.
Therefore, The model is robust and reliable because it exhibits a good balance between training performance and generalization ability.

Comparing Features Importance using SHAP AI

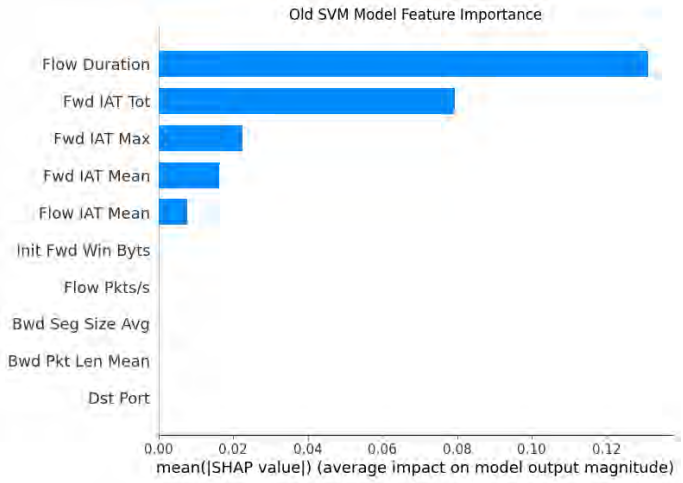

Figure 5.4: SVM Model Feature Importance

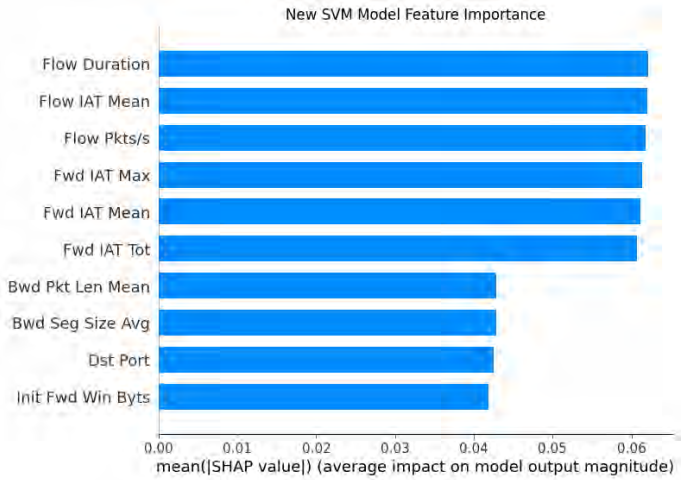

Figure 5.5: Improved SVM Model Feature Importance

Figure 5.4 represents the SVM models feature importance while figure 5.5 shows the improved SVM models feature importance using SHAP AI. While both SVM models give priority to "Flow Duration," the new model appears to give more weight to "Flow IAT Mean" and "Flow Pkts/s." This implies that the frequency and timing of packets within a flow may have a greater impact on the new SVM model. With multiple features making a substantial contribution, the new SVM model seems to have a more evenly distributed feature importance. The existing SVM model, on the other hand, appears to rely largely on a small number of important features. The improved SVM model performs better because it has discovered more subtle patterns in the data, as evidenced by the changes in feature importance.

Overall, the comparison of the two feature importance charts suggests that the new SVM model is more robust and capable of capturing a wider range of network traffic characteristics.

Serial No.	**Research Study**	**Testing Accuracy**
1	Akepalli et al., 2024 [39]	96.2%
2	Abdiyeva-Aliyeva., 2023 [32]	95%
3	Sharma et al., 2023 [38]	95%
4	Hdaib et al., 2024 [42]	97%
5	Proposed SVM Model	**99.46%**

Table 5.4: Comparison of Accuracy Across Studies

The table 5.4 presents a comparative analysis of testing accuracies from several research works in the fields of data analysis and machine learning. All of these research rely on the identification of data breaches on comparable datasets. Hdaib et al. (2024) demonstrated steady progress in this area with a remarkable accuracy of 97 percent. Importantly, the Proposed Optimised SVM Model achieved an exceptional accuracy rate of 99.46 percent, surpassing all previous studies. The model's advanced optimisation techniques, greater learning capabilities, and improved generalisation across similar datasets are all demonstrated by this result. The near-perfect accuracy of the Optimised SVM model demonstrates its suitability for real-world situations requiring accuracy and dependability. Our suggested model performs better than traditional methods by automating procedures like feature selection, parameter tweaking, and model optimization. This marks an important advance in the study of data breach attack detection.

5.1.3 Malware Results

Result Metrics

Metrics	RF	LR	NN	Optimized RF
Accuracy	0.9838	0.9249	0.9886	0.9942
Precision	0.9837	0.9304	0.9886	0.9883
Recall	0.9838	0.9249	0.9886	0.9927
F1 Score	0.9837	0.9219	0.9887	0.9905

Table 5.5: Performance Metrics

The table 5.5 shows that Optimized RF has the highest accuracy 98.86 percent and Logistic Regression has the lowest accuracy 92.49 percent. Optimized RF has the highest precision and recall for both, indicating that it makes fewer false positive and false negative predictions compared to the other models. Non-optimized Random Forest has a great recall and precision as well. With respect to the other models, the precision of 93.04 percent and recall of 92.49 percent are comparatively lower for Logistic Regression. This indicates uneven performance, suggesting the need for enhancements in the recall of detecting malware. The performance of the RF was improved by hyperparameter tuning.

Accuracy: With an accuracy of 0.9942, the Improved RF Model performs better than the non-optimized RF Model in terms of accuracy, precision, recall, and f1-score. This shows an increased accuracy in predictions overall by the Enhanced Model.

Precision: of the Improved RF Model is higher (0.9883) than the other Models. This implies that the Optimized RF Model works better at precisely identifying positive cases while preventing false positives.

Recall: The Improved RF Model has a higher recall rate of 0.9927 compared to 0.9838 for the other RF Model. This demonstrates the Improved RF Model's better capacity to detect accurate positive cases, increasing reliability for situations where identifying. All positive cases are crucial.

F1-Score: The Improved RF Model demonstrates a significant enhancement in the F1-Score, reaching 0.9905 in contrast to the non-optimized RF Model's 0.9837. This indicates an improved equilibrium between accuracy and completeness in the Improved Model.

The Improved RF Model shows clear improvements in Accuracy, Precision, Recall, and F1-Score, making it better suited for applications focusing on thorough detection

of positive cases. In general, the Improved RF Model offers a more equitable and strong performance.

Learning Curve

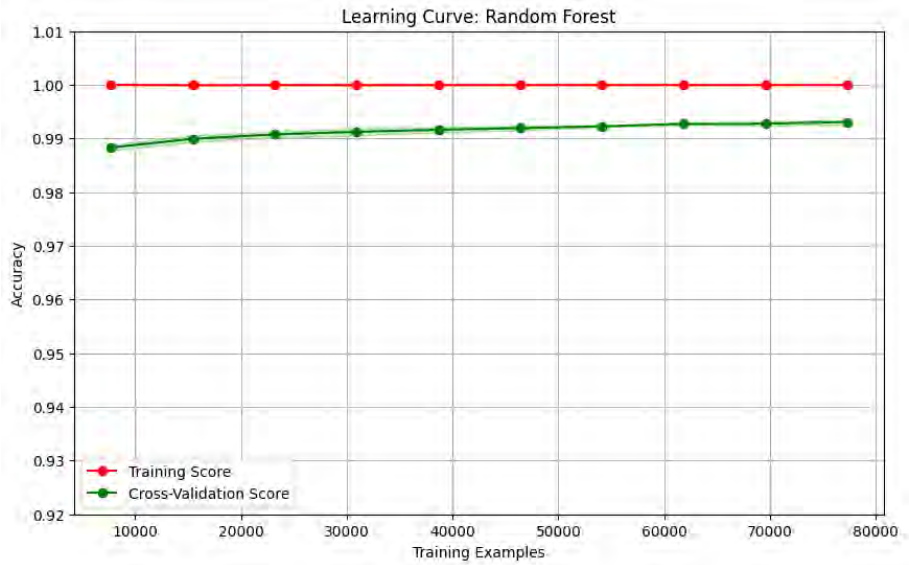

Figure 5.6: Leaning Curve of Optimized Random Forest Model

From the figure 5.6, the insights derived from the learning curve are as follows:

Training Score: The red line denoting the training score is positioned at or very near 1.0 (100 percent accuracy), signifying that the Random Forest model demonstrates nearly flawless performance on the training dataset.

Cross-Validation Score: The green line, which represents the cross-validation score, commences slightly lower but ascends and stabilizes at approximately 99 percent accuracy, closely mirroring the training score. This indicates the model exhibits strong generalization capabilities to unseen data with minimal overfitting.

Gap Between Curves: The model appears to be learning well as the number of training instances rises, as indicated by the little difference between the cross-validation and training scores.

Stability: Both curves maintain a stable trajectory without exhibiting significant fluctuations, indicating a robust model supported by adequate training data.

This learning curve reveals that the Optimized Random Forest model is performing remarkably well with strong generalization traits. The model is well-tuned and effective.

Comparing Features Importance using SHAP AI

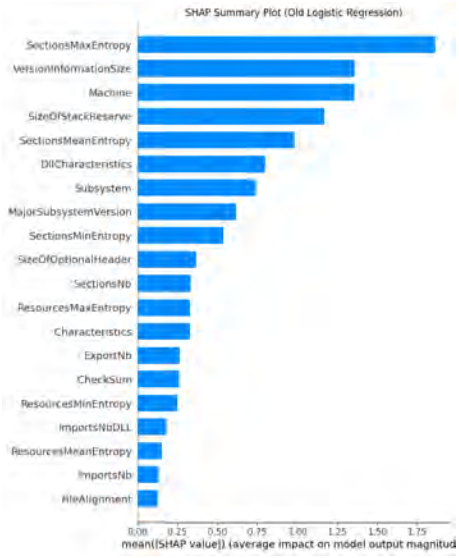

Figure 5.7: Non-optimized RF Models Feature Importance

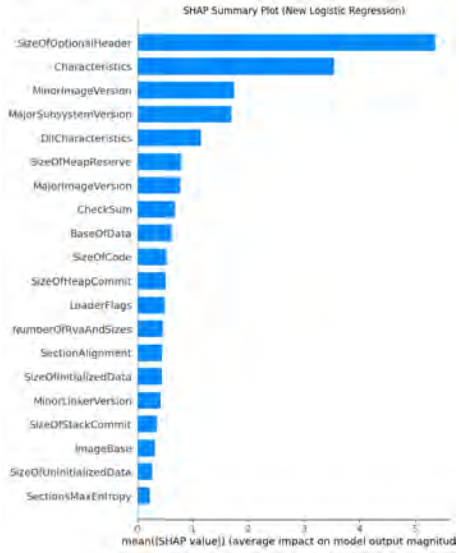

Figure 5.8: Opmitimized RF Models Feature Importance

The non-optimized model's feature importance is displayed in Figure 5.7, whereas the optimized model's feature importance utilizing SHAP AI is displayed in Figure 5.8. SectionsMaxEntropy, VersionInformationSize, and Machine are the most significant features in the current RF model, according to the SHAP Summary Plot. On the other hand, ImportsNb, FileAlignment, and ResourcesMeanEntropy have the least effect on the model's outcome.

SizeOfOptionalHeader, Characteristics, and MinorImageVersion are the most important aspects in the Improved RF model, according to the SHAP Summary Plot. In contrast, the least amount of effect on the model's output is had by SectionsMaxEntropy, SizeOfUninitializedData, and ImageBase.

Due to the PCA transformation, direct comparison of feature importance between the two plots is not possible.

The PCA treatment has improved the model's performance by removing noise and finding the most important underlying patterns in the data. Consequently, specific features in the enhanced model were shown to be important.

Comparative Analysis of Malware Detection

Serial No.	**Research Study**	**Testing Accuracy**
1	Fournier et al., 2021 [23]	91.5%
2	Babbar et al., 2023 [35]	93%
3	Ariffin et al., 2023 [34]	94%
4	Aguila, 2024 [40]	94.74%
5	Qiang et al., 2022 [30]	95.7%
6	Proposed RF Model	**99.42%**

Table 5.6: Comparison of Testing Accuracy Across Studies

The table 5.6 shows the testing accuracies from several studies in the domains of data analysis and machine learning are compared in the table. The detection of malware on similar datasets is the foundation of all these studies. In this regard, Qiang et al. (2022) showed consistent advancement with an impressive accuracy of 95.7 percent. Furthermore, the Proposed Optimised Random Forest Model outperformed all earlier research with an outstanding accuracy rating of 99.42 percent. This outcome demonstrates the model's advanced optimisation methods, increased learning capacity, and enhanced generalisation across comparable datasets. The Optimised Random Forest model's near-perfect accuracy shows that it is appropriate for real-world scenarios where accuracy and reliability are necessary. By automating processes such as feature selection, parameter tuning, and model optimisation, our proposed model outperforms conventional approaches. This is a major advance in the field of malware detection research.

5.1.4 Phishing Attack Results

Random Forest Classifier with Bayesian Optimization

Model	Precision	Recall	F1-Score	Accuracy
Random Forest	0.82	0.66	0.73	0.8514
Decision Tree	0.81	0.66	0.73	0.8479
Optimized Decision Tree	0.79	0.67	0.73	0.8462
Proposed Optimized RF	0.82	0.66	0.73	0.8521

Table 5.7: Model Performance Comparison

This table 5.7 shows the models performance metrics on phishing detection. Fine-tuning of the parameters in the Random Forest model, such as `n_estimators`, which refers to the number of trees, `max_depth`, which is the maximum depth for each tree, and several other important parameters, was achieved using Bayesian Optimization. With this optimization, the Random Forest Classifier improved to achieve an accuracy of 85.21%. This represents a notable improvement from an initial accuracy of 84.14% by a margin of 1.07%.

Making a probabilistic model of the goal function is how Bayesian optimisation operates, which in this case is the accuracy over the hyperparameter space, and iteratively selects which parameter settings to evaluate. It uses prior observations to model how different combinations of hyperparameters affect model performance, focusing computational efforts on the most promising combinations. This enables Bayesian Optimization to converge to the best hyperparameters much faster compared to other methods and, therefore, be computationally efficient while ensuring high performance.

This reflects an improved accuracy of 85.21%, which could indicate that Bayesian Optimization found a better combination of hyperparameters to allow the Random Forest Classifier to learn the patterns and generalize better across the dataset. Random Forest's ensemble learning benefits from the optimized decision tree configurations, now aligning better with the underlying patterns in the phishing detection problem. The features include `SSLfinal_State`, depicting the validity of the SSL certificate, `URL_Length`, and `having_IP_Address`; thanks to Bayesian Optimization, all of these are given more weight in the Random Forest model's selection procedure in order to find these patterns.

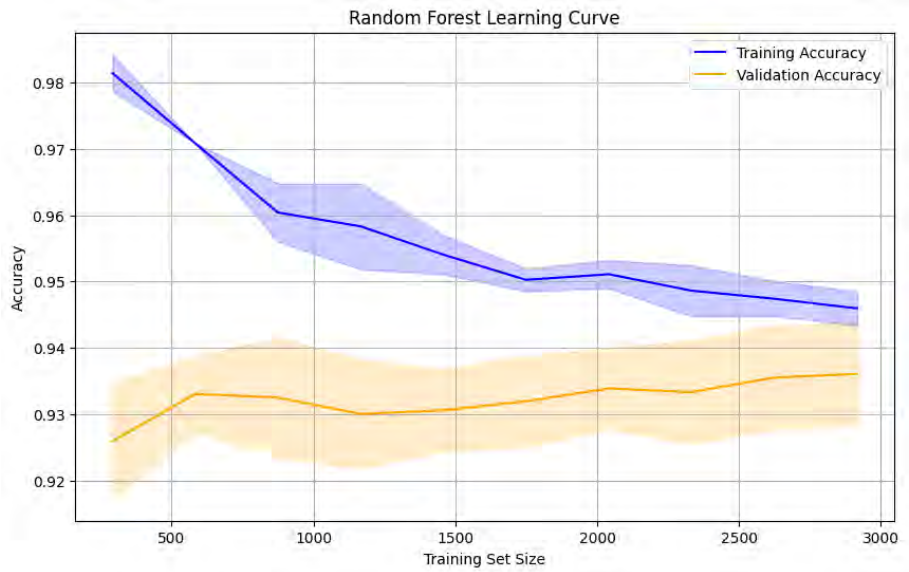

Figure 5.9: Learning curve for a Random Forest

Figure 5.9 is the learning curve for the decision tree which appears to converge at the end, indicating that the models performance is slowly getting stabilized. However, at first, with fewer samples the models tend to overfit but at the end, there is a comparatively small difference between the training and validation accuracy.

The ensemble's robustness is thus a factor in this regard. Because a random sample of the features and observations is used to train each decision tree, a Random Forest will never depend on just one feature or a sub-set of features. Bayesian Optimization ensures the selection of optimum number of trees along with other parameters which enhances model performance by allowing better generalization across different types of phishing patterns observed in the test set.

Optimized Decision Tree

The Decision Tree Classifier initially obtained an accuracy of **84.79%** on unseen test data in the context of the phishing detection problem using the given dataset. According to this baseline performance, the model was able to identify trends in the dataset and extrapolate them to predict whether a website was phishing or authentic. The features such as **SSLfinal_State**, **URL_Length**, and **having_IP_Address** were key indicators that Decision Trees identified and used to make these predictions.

The initial Decision Tree, without advanced hyperparameter tuning, achieved an accuracy of **84.79%** out of the box. This value indicates that the Decision Tree was doing well in finding meaningful patterns in the data that distinguish phishing sites from legitimate ones. Features such as **SSLfinal_State**, **URL_Length**, and **having_IP_Address** were some of the critical indicators identified by the Decision Tree in correctly classifying the websites. Despite the high degree of accuracy, by

their nature, Decision Trees are sensitive to fluctuations in training data and may result in overfitting if their hyperparameters are not fine-tuned.

The initial performance gave some indication that the Decision Tree was learning feature-target variable relationships through thresholding data on key features in splits-for example, two large tells for phishing: valid SSL certificates or suspiciously long URLs-and it could utilize those to do its predictions.

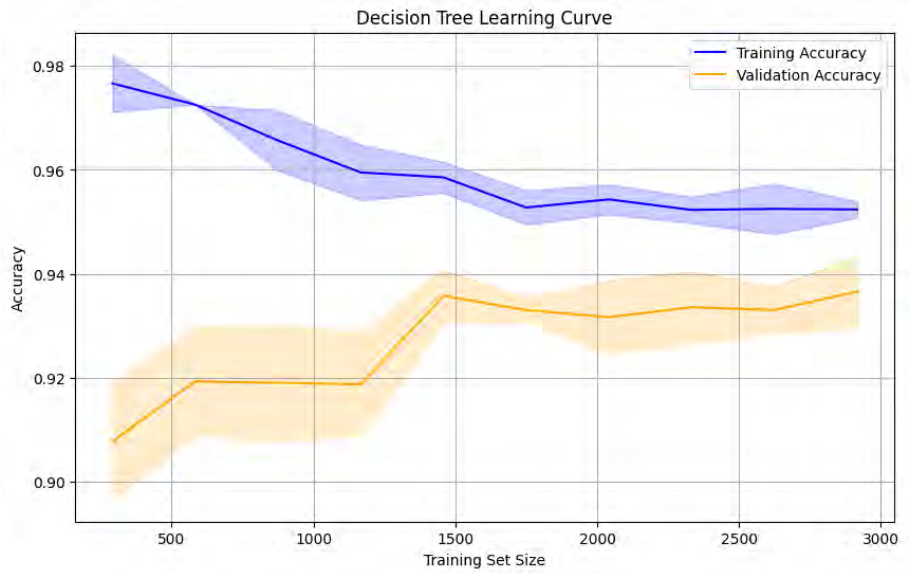

Figure 5.10: Learning Curve for a Decision Tree classifier

Figure 5.10 is the learning curve for the decision tree which appears to converge at the end, indicating that the models performance is slowly getting stabilized. However, at first, with fewer samples the models tend to overfit but at the end, there is a comparatively small difference between the training and validation accuracy indicates that is not overfitting and is effectively generalizing to new data.

Some drawbacks with Decision Trees are that they can face overfitting when deep trees are concerned. When this happens, generalizable patterns are not picked by the model; it instead captures noise in the training data. Maximum depth, minimum samples per split, and minimum samples per leaf all work as hyperparameters towards tuning the model during the time of training to be neither overly simple nor too complex. Looking at the learning curve it overfits at first, however, the gap between the training score and validation score decreases.

Effect of Hyperparameter Tuning

To optimize performance and address potential overfitting, hyperparameter tuning was applied to the Decision Tree model. Hyperparameter tuning involves systematically adjusting the model's parameters to find the best combination for accuracy and generalization. The parameters commonly tuned in Decision Trees include:

- **max_depth (Maximum Depth):** The maximum depth of the decision tree.
- **min_samples_split:** The bare minimum of samples needed to split a node.
- **min_samples_leaf:** The bare minimum of samples required for a leaf node.
- **max_features:** Restricts how many features are taken into account at each split.

Typically, **Bayesian Optimisation** or **Grid Search** are used to determine the optimal set of these parameters. Having done that with these hyperparameters, the model's performance was re-evaluated to have a test accuracy of **84.62%**, slightly lower than the initially reported **84.79%**. Although this could be considered a reduction, the implication of such a result means that its generalization improved, resulting in less overfitting. It should therefore perform well with consistency on data it hadn't seen before.

Why Did the Accuracy Drop? That would be justified by the slight drop from **84.79%** to **84.62%** after hyperparameter tuning, as the nature of this tuning often restricts the model's complexity–for instance, restricting the tree's maximum depth or giving **min_samples_split** or **min_samples_leaf** greater values. In this respect, these adjustments will avoid the learning of noise–the too specific patterns–by the Decision Tree. Although these constraints result in a minor loss in training performance, it makes the Decision Tree generalize better on new, unseen data.

Overfitting: If not constrained properly, Decision Trees can easily overfit the training data and capture noise, hence leading to poor generalization. Model Simplicity: A tuning parameter to avoid overfitting. It simplifies the tree, reducing a bit of performance over the training data, although it usually generalizes much better on test data. **Bias-Variance Tradeoff**: The concept of bias and variance in hyperparameter tuning is also relevant. In forcing the model to fall into a specific complexity, the Decision Tree introduces higher bias with lower variance for better generalization. These results bring out a point–that **hyperparameter tuning** is not about test accuracy maximization but also about keeping the model robust and not overfitting. A decrease in test accuracy from **84.79%** to **84.62%** after tuning shows that the Decision Tree has now become more focused on learning the most generalizable patterns rather than fitting every detail in the training data.

5.2 Discussion

This study focuses on optimization of machine learning (ML) and artificial intelligence (AI) algorithms aimed at detecting and mitigating four significant cybersecurity threats: Distributed Denial of Service (DDoS) attacks, data breaches, malware, and phishing. Each threat was examined individually, and customized ML models were utilized to achieve optimal outcomes. Our results underscore the effectiveness of ML/AI in accurately identifying these threats, highlighting their potential for application in real-world scenarios.

For DDoS attack detection, we attained an impressive accuracy rate of 99.99%, utilizing the TPOT evolutionary algorithm. This approach enabled automated model optimization, ensuring the identification of the most effective pipeline for this specific application. The nearly perfect accuracy reflects the robustness of the methodology in recognizing patterns indicative of DDoS attacks.

In the realm of data breach detection, we applied a Support Vector Machine (SVM) model, achieving an accuracy of 99.46%. This finding illustrates the SVM's proficiency in managing high-dimensional data while detecting subtle anomalies linked to potential breaches. The model's performance further emphasizes its suitability for identifying complex patterns associated with data exfiltration or unauthorized access.

By using the Random Forest (RF) technique, we were able to detect malware with an accuracy rate of 99.94%. The RF classifier proved to be exceptionally effective, adept at identifying intricate data relationships while maintaining general applicability. This achievement demonstrates the algorithm's strong capability in differentiating between benign and malicious activities.

Conversely, phishing detection yielded a lower accuracy of 85.21%, also utilizing the Random Forest algorithm. Although the RF model delivered reasonable performance, phishing detection remains a challenging endeavor due to the subtle and rapidly evolving nature of phishing techniques. This relatively lower accuracy indicates opportunities for enhancement, possibly through the adoption of ensemble methods, deep learning techniques, or broader datasets that encompass a wider array of phishing scenarios.

Various ML models were in the experimentation phase. Through iterative tuning and evaluation, we confirmed that the selected models achieved the best balance between performance and computational efficiency for each specific task. Overall, this research illustrates the promising capabilities of ML/AI-driven methodologies in addressing critical cybersecurity challenges. The high accuracy rates obtained for DDoS, data breach, and malware detection indicate these models are primed for deployment in practical applications. Meanwhile, the outcomes for phishing detection suggest a need for additional refinement and exploration of complementary techniques. Future endeavors could concentrate on enhancing phishing detection accuracy by integrating advanced methodologies.

5.3 Constraints

While the work presented in this thesis demonstrates significant improvements in detecting data breaches and malware using hyperparameter-tuned models, there are certain limitations that must be acknowledged. These limitations highlight areas for further refinement and provide context for the results achieved.

Dependence on Data Quality and Availability: The quantity and quality of the training data are directly linked to the effectiveness of the decision tree-based optimized phishing detection system. An unbalanced dataset with few samples might have hindered the model's capacity to generalize successfully in the context of phishing detection.

Computational Overheads: Hyperparameter tuning was conducted using GridSearchCV, which systematically evaluates all combinations of specified hyperparameters. While this approach guarantees finding the optimal parameters within the defined search space, it is computationally expensive, especially for complex models like SVM with non-linear kernels. The exhaustive nature of grid search may have limited the exploration of larger or more diverse hyperparameter spaces due to resource constraints.

Scalability: The tuned models performed well on the given dataset; however, their scalability to larger datasets or real-time systems was not thoroughly evaluated. For example, the SVM model, particularly with a non-linear kernel, may face challenges in handling high-dimensional or large-scale data due to increased computational complexity.

Chapter 6

Conclusion

As cyber-attacks are increasing day by day, the best possible approach is to prevent cyber attacks by identifying the vulnerabilities, loopholes, and weaknesses of systems, as well as taking appropriate preparation against those attacks. However, cyber attacks can come in new forms every day, and even a proper up-to-date defense mechanism can be a victim of zero-day vulnerability. By incorporating AI into pen-testing, our research has proposed a hybrid method to effectively defend these threats. Threat modeling using AI can be more efficient and can help organizations get results more quickly by helping them understand the security contexts and outputs more quickly. We demonstrated this in our model implementation part, where it can be seen that different AI models such as Random Forest, Neural Networks, KNN etc. can detect anomalies in data and intrusion in systems with a very high accuracy. However, regular versions of SVM and RF in some cases can be seen as having comparatively low accuracy. To tackle this, we have proposed improvements such as hyperparameter tuning, feature selection, feature engineering with PCA, and evolutionary algorithms for ML models with lower accuracies to improve their prediction accuracies. This significantly improved their ability to detect threats. We not only obtained betterment compared to the original models' metrics but also obtained betterment compared to the existing modification of those models in terms of detecting anomalies. This ensures that all the major machine learning models can have a high accuracy rate in terms of detecting Using AI in vulnerability analysis can help to continuously monitor and detect threats to provide real-time incident response. In addition to this, we proposed the best-suited model with the most optimum performance for each of the threats. Incorporating AI with OPSEC's pen-testing, not only paves the way for automating the effectiveness of OPSEC but also reduces the chances of human error. By analyzing past data and security incidents, AI can detect threats and take proactive steps to resolve them before they are exploited.

6.1 Future Works

Despite attaining good progress in terms of improving ML models with comparatively lower accuracies in detecting vulnerabilities, obstacles remain in enhancing the detection models for phishing attacks. In the future, we aspire to explore possible ways to improve these areas, as well as to explore the chances to enhance and expand the overall threat detection framework. To do that, firstly, we can combine and hybridize different ML models. By merging rule-based systems with deep learning structures, the hybridization of machine learning approaches can improve the accuracy of threat detection, especially in phishing attacks. Secondly, our future work should be able to adjust to evolving threats i.e. our future research should prioritize developing flexible and evolving models that can learn from fresh data. Additionally, the Integration of threat intelligence can enhance the detection of new or emerging threats, especially for phishing attacks. Lastly, we need to prioritize the assessment of real-world systems. Although our study refines detection models in a controlled environment, future research should assess their resilience and expandability in real-world settings, considering practical issues like latency and compatibility.

Bibliography

[1] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995, vol. 4.

[2] S.-H. Liao, "Expert system methodologies and applications—a decade review from 1995 to 2004," *Expert systems with applications*, vol. 28, no. 1, pp. 93–103, 2005.

[3] P. Buchanan, B. Ramsay, A. Smales, and G. Russell, "Teaching penetration testing and malware analysis within a cloud-based environment," in *In Workshop on Cybersecurity Training & Education (VIBRANT15)*, 2015, p. 7.

[4] S. Meliopoulos, G. Cokkinides, R. Beyah, S. Walters, and P. Myrda, "Cyber security and operational reliability," in *2015 48th Hawaii International Conference on System Sciences*, IEEE, 2015, pp. 2655–2663. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2015.320.

[5] J. Dawson and J. T. McDonald, "Improving penetration testing methodologies for security-based risk assessment," in *2016 Cybersecurity Symposium (CYBERSEC)*, IEEE, 2016, pp. 51–58. [Online]. Available: https://dx.doi.org/10.1109/CYBERSEC.2016.016.

[6] A. Amos-Binks, J. Clark, K. Weston, M. Winters, and K. Harfoush, "Efficient attack plan recognition using automated planning," in *2017 IEEE symposium on computers and communications (ISCC)*, IEEE, 2017, pp. 1001–1006.

[7] D. Dalalana Bertoglio and A. F. Zorzo, "Overview and open issues on penetration test," *Journal of the Brazilian Computer Society*, vol. 23, pp. 1–16, 2017. [Online]. Available: https://doi.org/10.1186/s13173-017-0051-1.

[8] L. Wang, S. Jajodia, A. Singhal, A. Singhal, and X. Ou, *Security risk analysis of enterprise networks using probabilistic attack graphs*. Springer, 2017.

[9] B. Han, X. Yang, Z. Sun, J. Huang, and J. Su, "Overwatch: A cross-plane ddos attack defense framework with collaborative intelligence in sdn," *Security and Communication Networks*, vol. 2018, no. 1, p. 9 649 643, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1155/2018/9649643.

[10] S. Khan and S. Parkinson, "Review into state of the art of vulnerability assessment using artificial intelligence," *Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach*, pp. 3–32, 2018.

[11] S. Khan and S. Parkinson, "Review into state of the art of vulnerability assessment using artificial intelligence," *Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach*, pp. 3–32, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-92624-7_1.

[12] B. Berendt, "Ai for the common good?! pitfalls, challenges, and ethics pen-testing," *Paladyn, Journal of Behavioral Robotics*, vol. 10, no. 1, pp. 44–65, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1810.12847.

[13] D. R. McKinnel, T. Dargahi, A. Dehghantanha, and K.-K. R. Choo, "A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment," *Computers & Electrical Engineering*, vol. 75, pp. 175–188, 2019. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2019.02.022.

[14] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine-(asvm-) based detection for distributed denial of service (ddos) attack on software defined networking (sdn)," *Journal of Computer Networks and Communications*, vol. 2019, no. 1, p. 8 012 568, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1155/2019/8012568.

[15] A. M. Sandar, Y. Min, and K. M. N. Win, "Fundamental areas of cyber security on latest technology," *Published in International Journal of Trend in Scientific Research and Development (ijtsrd)*, vol. 3, no. 5, 2019. [Online]. Available: https://doi.org/10.31142/ijtsrd26550.

[16] H. J. Hejase, H. F. Fayyad-Kazan, and I. Moukadem, "Advanced persistent threats (apt): An awareness review," *Journal of Economics and Economic Education Research*, vol. 21, no. 6, pp. 1–8, 2020. [Online]. Available: http://dx.doi.org/10.13140/RG.2.2.31300.65927.

[17] A. Ibrahim, D. Thiruvady, J.-G. Schneider, and M. Abdelrazek, "The challenges of leveraging threat intelligence to stop data breaches," *Frontiers in Computer Science*, vol. 2, p. 36, 2020.

[18] I. Ko, D. Chambers, and E. Barrett, "Self-supervised network traffic management for ddos mitigation within the isp domain," *Future Generation Computer Systems*, vol. 112, pp. 524–533, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X20302193.

[19] J. A. Perez-Diaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning," *IEEE Access*, vol. 8, pp. 155 859–155 872, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9177002.

[20] S. Zeadally, E. Adi, Z. Baig, and I. A. Khan, "Harnessing artificial intelligence capabilities to improve cybersecurity," *Ieee Access*, vol. 8, pp. 23 817–23 837, 2020.

[21] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated ddos attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, p. 103 108, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1084804521001296.

[22] M. J. H. Faruk, H. Shahriar, M. Valero, *et al.*, "Malware detection and prevention using artificial intelligence techniques," in *2021 IEEE international conference on big data (big data)*, IEEE, 2021, pp. 5369–5377. DOI: 10.1109/BigData52589.2021.9671434. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9671434.

[23] A. Fournier, F. El Khoury, and S. Pierre, "A client/server malware detection model based on machine learning for android devices," *IoT*, vol. 2, no. 3, pp. 355–374, 2021. [Online]. Available: https://www.mdpi.com/2624-831X/2/3/19.

[24] J. Kinyua and L. Awuah, "Ai/ml in security orchestration, automation and response: Future research directions.," *Intelligent Automation & Soft Computing*, vol. 28, no. 2, 2021. [Online]. Available: https://doi.org/10.32604/iasc.2021.016240.

[25] I. H. Sarker, M. H. Furhad, and R. Nowrozy, "Ai-driven cybersecurity: An overview, security intelligence modeling and research directions," *SN Computer Science*, vol. 2, no. 3, p. 173, 2021.

[26] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "A survey on ethical hacking: Issues and challenges," *arXiv preprint arXiv:2103.15072*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2103.15072.

[27] B. A. Bin Arfaj, S. Mishra, and M. Alshehri, "Efficacy of unconventional penetration testing practices.," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, 2022. [Online]. Available: https://doi.org/10.32604/iasc.2022.019485.

[28] S. tul Hassan, "Analysis of vulnerabilities in system by penetration testing," *Pakistan Journal of Scientific Research*, vol. 2, no. 1, pp. 22–25, 2022. [Online]. Available: https://doi.org/10.57041/pjosr.v2i1.23.

[29] B. R. Maddireddy and B. R. Maddireddy, "Ai-based phishing detection techniques: A comparative analysis of model performance," *Unique Endeavor in Business & Social Sciences*, vol. 1, no. 2, pp. 63–77, 2022. [Online]. Available: https://unbss.com/index.php/unbss/article/view/43.

[30] W. Qiang, L. Yang, and H. Jin, "Efficient and robust malware detection based on control flow traces using deep neural networks," *Computers & Security*, vol. 122, p. 102 871, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167404822002656#:~:text=The%20proposed%20classifier%20can%20effectively,handling%20the%20evolution%20of%20malware.

[31] X. Zhang, M. M. Yadollahi, S. Dadkhah, H. Isah, D.-P. Le, and A. A. Ghorbani, "Data breach: Analysis, countermeasures and challenges," *International Journal of Information and Computer Security*, vol. 19, no. 3-4, pp. 402–442, 2022.

[32] G. Abdiyeva-Aliyeva, "Ai-based network security anomaly prediction and detection in future network," in *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, 2023, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10131845.

[33] E. A. Altulaihan, A. Alismail, and M. Frikha, "A survey on web application penetration testing," *Electronics*, vol. 12, no. 5, p. 1229, 2023. [Online]. Available: https://doi.org/10.3390/electronics12051229.

[34] N. A. M. Ariffin and H. P. Casinto, "Android malware detection using permission based static analysis," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 33, no. 3, pp. 86–97, 2023. [Online]. Available: https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/article/view/2291.

[35] H. Babbar, S. Rani, D. K. Sah, S. A. AlQahtani, and A. Kashif Bashir, "Detection of android malware in the internet of things through the k-nearest neighbor algorithm," *Sensors*, vol. 23, no. 16, p. 7256, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/16/7256.

[36] M. Markevych and M. Dawson, "A review of enhancing intrusion detection systems for cybersecurity using artificial intelligence (ai)," in *International conference Knowledge-based Organization*, vol. 29, 2023, pp. 30–37. [Online]. Available: https://doi.org/10.2478/kbo-2023-0072.

[37] M. A. Al-Shareeda, S. Manickam, and M. A. Saare, "Ddos attacks detection using machine learning and deep learning techniques: Analysis and comparison," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 2, pp. 930–939, 2023. [Online]. Available: https://doi.org/10.11591/eei.v12i2.4466.

[38] K. Sharma, M. Chaudhary, K. Yadav, and P. Thakur, "Anomaly detection in network traffic using deep learning," in *2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET)*, 2023, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10419951.

[39] S. Akkepalli and K. Sagar, "Anomaly-based network intrusion detection using hybrid cnn, bi-lstm deep learning techniques," in *2024 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10548678.

[40] R. Baker del Aguila, C. D. Contreras Pérez, A. G. Silva-Trujillo, J. C. Cuevas-Tello, and J. Nunez-Varela, "Static malware analysis using low-parameter machine learning models," *Computers*, vol. 13, no. 3, p. 59, 2024. [Online]. Available: https://www.mdpi.com/2073-431X/13/3/59.

[41] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware detection with artificial intelligence: A systematic literature review," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–33, 2024. [Online]. Available: https://doi.org/10.1145/3638552.

[42] M. Hdaib, S. Rajasegarar, and L. Pan, "Quantum deep learning-based anomaly detection for enhanced network security," *Quantum Machine Intelligence*, vol. 6, no. 1, p. 26, 2024. [Online]. Available: https://link.springer.com/article/10.1007/s42484-024-00163-2.

[43] S. S. Lumpatki and S. Patwardhan, "An overview of artificial intelligence applications in cybersecurity domains," in *International Conference on Smart Computing and Communication*, Springer, 2024, pp. 11–24. [Online]. Available: https://doi.org/10.1007/978-981-97-1326-4_2.

[44] P. Robinson, *The MOVEit Attack Explained — lepide.com*, https://www.lepide.com/blog/the-moveit-attack-explained/, 2024.