ROBERT GORDON UNIVERSITY ABERDEEN

SCHOOL OF COMPUTING

MSc Project Report

Student Name: Timileyin S Falomo	Matriculation Number:
Supervisor: Dr. Andrei Petrovski	Second Marker: Dr. M S Mekala
Course: MSc Cyber Security	
Project Title: Improved Detection of HTTPS Malware Traffic without Decrypting	
Start Date: 06/02/2023	Submission Date: 22/04/2023

CONSENT

I agree

That the University shall be entitled to use any results, materials or other outcomes arising from my project work for the purposes of non-commercial teaching and research, including collaboration.

DECLARATION

I confirm:

- **That the work contained in this document has been composed solely by myself and that I have not made use of any unauthorised assistance.**

- **That the work has not been accepted in any previous application for a degree.**

- **All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.**

Student Signature: Timileyin S Falomo	Date Signed: 22/04/2023

AKNOWLEDGEMENT

I want to express my gratitude to my supervisor, Dr. Andrei Petrovski, for his unending support, tolerance, and competent guidance while working on this project report. He provided additional support and made time outside of our scheduled activities to guide me through the areas where I encountered difficulties, therefore I consider myself very fortunate to have been under his supervision.

I would especially like to thank all the lecturers at Robert Gordon University, for providing me with all the assistance I needed to complete my master's degree successfully. You all are amazing, and the learning materials available are outstanding.

Finally, I want to thank my wife for her encouragement and support during my project and for her desire to see me succeed.

ABSTRACT

Encrypted traffic data has been a key technique used to evade existing malware detection techniques. Deep packet inspection techniques that can be used to detect malware affect user privacy as the packets must be decrypted before the detection process. These attacks cost individuals and organizations around the world both financially and in terms of their reputation. This project utilizes a machine learning approach with a fully encrypted dataset called CIRA-CIC-DoHBrw-2020 to provide a highly effective solution to malware traffic detection without decrypting or affecting privacy for data in transit. The results from the project showed 99.9% accuracy, confirmed by a recall score of 0.986, a precision score of 0.995, an f1-score of 0.991, and a confusion matrix with 0.03% false positives and 0.1% false negatives.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Any software that is meant to carry out damaging and malicious deeds on a computer system is referred to as malware, which is the abbreviation for "malicious software." Spyware, rootkits, worms, Trojan horses, logical bombs, and viruses are examples of malware. These malwares come in a variety of shapes and sizes, from specifically designed system attacks to standardised self-replication probes that attack accessible targets. (Wangen 2015).

The primary cryptographic technique for preserving data integrity and privacy during inter-party transmission in the current Internet context is HTTPS. Different user types are engaged with HTTPS, such as administrators who must deal with complicated compatibility concerns and end users who must make critical security decisions when prompted by warnings. (Krombholz et al. 2019). The Secure Sockets Layer (SSL) or Transport Layer Security (TLS) is used by HTTPS to create an encrypted channel of communication between servers and clients.

The Internet Engineering Task Force (IETF) adopted the TLS standard in 1999, and it is mostly used to transform HTTP to HTTPS. (Shbair et al. 2015). The two core parts of TLS are the Record Protocol, which provides a secure channel for managing data transmission, and the Handshake Protocol, which handles key formation and authentication. Application data is encrypted using application keys produced during the Handshake Protocol and utilised throughout the Record Protocol. (Krawczyk, Paterson and Wee 2013). TLS and SSL, two cryptographic technologies, provide privacy and integrity. Unfortunately, security and privacy are not synonymous, and threat actors are exploiting TLS to mask illegal actions including Command and Control, malware installation on a network, and data theft.

This project aims at detecting encrypted malicious traffic using a machine learning approach as HTTPS and encrypted traffic is being used to evade detection from existing mitigation techniques.

1.2 Motivation for Work

In my home country Nigeria, government agencies, financial institutions, the energy sector, education, health and other very sensitive sectors are gradually migrating to be fully dependent on computing-based solutions. Recent studies have shown that nations that depend on computing-based solutions and devices have in recent times seen an increased amount of sophisticated attacks especially using encrypted traffic to evade detection. I was motivated to do this research to build a non-traditional and non-signature-based highly effective solution that can mitigate these threats to governments, organizations and groups of individuals as the traditional methods have failed. The solution to this if implemented in Nigeria will provide a proactive approach to these attacks and limit the attacks success rate which will reduce cost and organizational reputation loss.

1.3 Content of the Rest of the Project

The remaining part of this project is structured into chapters and its contents are presented below.

1. Chapter Two- Literature Review: A review of previously completed research and publications. This chapter reviews earlier technical work, critically examining the issues resolved and advancements made as a consequence of the study that provided the intended outcomes.
2. Chapter Three- Project Specification: This chapter presents the methodology utilized in meeting the research goals, the functional and non-functional requirements, the legal, ethical, social and environmental issues as well as code of practice and industry standards.
3. Chapter Four- Design Alternatives and Justification for Chosen Design: This chapter presents the machine learning design approach alternatives, the justification for the selected waterfall approach, alternative approaches to encrypted traffic detection and the justification for the choice of machine learning in encrypted malware traffic detection.
4. Chapter Five- Implementation and Testing: This chapter presents the system requirements, research results and the implementation of the solution methods highlighted in Chapter three. This chapter also presents the testing and evaluation process.
5. Chapter Six- Evaluation of Work: This chapter fully explains the experiment's outcome and the research and analysis of the impact on the industry. This chapter also evaluates outcome with existing literature, the projects objectives, the functional requirements and the contribution to knowledge.
6. Chapter Seven- Conclusion and Future Work: This is the final chapter of the project report. This chapter highlights the strides toward meeting the project aim. The outcome

of the research is also highlighted as well as recommendations. A section on future work and possible improvements about the work are also presented.

CHAPTER 2: LITERATURE REVIEW

This chapter presents the review of existing and related literature on HTTPS malware traffic detection. This chapter also presents the research gap, existing solutions and the mitigation effects, and the building of the theoretical framework for this project.

2.1 What is a Malware?

Malware which is the short form of "malicious software" is referred to any type of software intended to perform malicious and harmful acts on a computer system. Malware includes spyware, rootkits, worms, Trojan horses, logical bombs, and viruses. These malwares can take many different forms, from specially created attacks on systems to generic self-replication probes that attacks available targets (Wangen 2015).

Alrammal et al. (2022) defined malware as a code or set of instructions used to obtain access to confidential information and computing devices. Deng et al. (2023) refer to malware as a type of software that aims to bring harm or exploit an operational process of a computing device. These three definitions of malwares can be summarized as a software created with the primary purpose of bringing harm to a computing device. The next section presents the researches attempts on the categorization of malwares.

2.1.1 Categories of Malware

With the high number of malware cases estimated at 1 billion in 2022 alone, categorizing malwares is quiet challenging for researches. The categorization is important for better detection and mitigation of a particular family or type of malware (Deng et al. 2023). Malware substantial threat to information security and can categorized into viruses, trojan horses, ransomware, rootkits, worms and zero-day amongst others as malware variants are on the steady increase. (Aboaoja et al. 2022)

Li et al. (2022) categorized malwares into four types comprising of trojan, virus, ransomware and worms. They also attempted to categorized malwares based on their physiology using machine learning techniques and malware detection.

Tayyab et al. (2022) categorized malwares into 9 types namely worms, viruses, backdoor, trojan horses, botnets, spyware, downloader, rootkit and scareware amongst others. According to Tayyab et al. (2022), malwares can fall under one or more categorizes based on their characteristics.

According to Kara (2022), malwares can be categorized into two major parts fileless malwares and host dependent malwares. Fileless malwares are also referred to as host independent malwares. Due to the nature of fileless malwares it is difficult to detect even with modern malware detection techniques. Fileless malware attack consist of four stages delivery method, code injection, persistence and execution. The foregoing section presents the seriousness of malware attacks.

2.1.2 Seriousness of Malware Attacks

In 2021, the global damage cost of successful malware is estimated at 6 trillion US dollars. Malwares have the potential to cause great damage to information systems. A 22.9% increase in development malwares have been recorded in recent years (Aboaoja et al. 2022).

According to Li et al. (2022), over 114 million malwares are developed each year with 78% targeted at windows machines. Ransomware a malware type has be tagged a very dangerous which has ruined both small and large business including rail companies in Europe, Nissan and NHS organizations amongst others. Companies spend an estimate of 2.4 million US dollars to prevent and detect these attacks. Stuxnet is a popular malware used against the Iranian nuclear plant sending their research years behind (Tayyab et al. 2022). The table below provides information of some recent malware attacks and the cost of the attack.

Table 2.1 Recent Malware Attacks and the Costs (Guerrero-Saade 2022, Katagiri 2022, Stejskal and Faix 2022)

FIRM AFFECTED	YEAR	CHARACTERISTICS	INFORMATION AFFECTED
Ukraine Organizations	2022	Deployment of a wiper malware targeted at Windows devices accompanied by PartTicket Spyware. The malware is called HermeticWiper.	Widespread DDOS attack for a week and operational sabotage
PyeongChang Olympic Games	2018	Compromised service providing companies in order to gain access. Olympic Destroyer (malware)	Reservation information, paralyzed information systems.
Sejong	2017	ActiveX Zero-Day, PinLady's plugin	

Institute (South Korea)		detect	
Polish and Bangladesh Banks	2016	RATANKBA malware (exploited watering hole)	
RSA	2011	Phishing mail (exploited an Adobe flash vulnerability) PoisonIvy malware (zero day)	Stole customer bank authentication details
Sony	2011	Drive-by download Vulnerability in Internet explorer Aurora malware (zero-day)	Customers authentication details, Personal Identifiable Information and Confidential information
IRAN NUCLEAR PLANT	2011	Kernel level exploit, Vulnerability: win32k.sys Stuxnet malware (zero-day)	Confidential information on plans for Nuclear warhead

Table 2.1 presents seven malware attacks that have occurred, the attack method and the affected resources. This is a few out of the so many malware attacks. The challenge in getting cyberattack details and information about assets affected is due to privacy, company reputation and legal issues. Organizations are also scared of divulging sensitive information while reporting attacks (Doriguzzi-Corin and Siracusa 2022).

The Ukrainian organizations attacked by the HermeticWiper malware were hit using a fraudulent digital certificate issued by Hermetica Digital Ltd. This certificate has since disappeared after these attacks. This attack was discovered in February of 2022 by threat intelligence. This attack is one in so many attacks recorded in 2022 (Guerrero-Saade 2022).

According to Kim and Kim (2019), The PyeongChang Olympic 2018 attack targeted the organizing committee server. This attack has been linked to different countries including china, Russia and North Korea although, code snippets obtained post the attack revealed that a north Korean group Lazarus was involved. The financial implication of the PyeongChang has not been made public.

Other attacks presented in Table 2.1 have also shown the seriousness of these attacks and the effects on organizations. These attacks have used different malwares which fall under multiple malware families. These attacks are spread over a period of 12 years and affecting popular organizations. The loss due to these attacks range from denial of service to loss of personal and confidential data.

2.1.3 Vulnerabilities Exploited by Malware

Malwares usually succeed because vulnerabilities exist in targeted machines (Kumar and Lim 2019). All identified vulnerabilities have been captured with unique identities in the Common Vulnerabilities and Exposures (CVE) database. The CVE database also classifies and provides the vulnerability details and possible mitigation techniques (Mitre Corporation 2021). The risks of exploiting any of these attacks are also provided in the Nation Vulnerability Database (NVD) managed by the National Institute for Standards and Technology (NIST) (Byers, Waltermire and Turner 2020)

Table 2.2 Identified vulnerabilities for different devices (Kumar and Subbiah 2022)

S/No	**Device Vendor**	**Vulnerabilities discovered**
1	Microsoft Windows	6646
2	Linux Distribution	10832
3	Apple	5433
4	Android	3875

These vulnerabilities identified in Table 2.2 is also based on existing software's. The table shows the number of known vulnerabilities with reference to device vendors but unknown vulnerabilities still exist (Kumar and Subbiah 2022).

A common vulnerability in androids is users installing and granting permissions to fake apps. This vulnerability was largely used during the COVID period to exploit devices (Manzil and Naik 2022). CVE-2021-0306 identifies a vulnerability associated with android devices that can bypass and agree with all permissions. The risk is categorized as high and can allow an adversary get escalated privileges (Mitre Corporation 2021).

With applications constantly being created to accommodate human needs and to make living in different stages of life easier (Asongu and Le Roux 2017), some of the developers of these applications have malevolent intentions. These malicious programmes cannot be classified as trustworthy or untrusted at the time of their creation because they are unclear. These malwares are classified as "Zero-day" which exploit "Zero-day" vulnerabilities because they are obscure and contain malicious code (Kumar and Subbiah 2022, Zhang et al. 2017). Zero-day malwares are unique because they are difficult to identify at the time of creation and deployment because no security tool has yet identified them as the vulnerability exploited remains or in existence (Nicho, Oluwasegun and Kamoun 2018).

CVE-2018-0907 is the unique identifier for this issue. A new vulnerability known which exploits VBA macros and DDE. An adversary gains access through the user’s weakness in downloading MS word files which might come with a malicious attachment. This malware vulnerability is of great concern as windows has the highest population of subscribers worldwide (Koutsokostas et al. 2022).

Double extensions is a vulnerability that involves hiding an executable file with another suffix, typically .doc or .pdf extension. This is also known as a hidden executable programme and is referenced in the common vulnerabilities and exposures data base as CVE-2020-13671 with a risk score of 8.8 (Byers, Waltermire and Turner 2020, Mitre Corporation 2021, Paik et al. 2022).

Malware that can change its form frequently in order to avoid detection is known as Malware metamorphosis or obfuscation, and this is a rising trend. The application's properties are altered by this technique at time intervals to evade detection. Security tools that rely on signatures to identify viruses cannot keep up with the constant changes to software. As of 2017, methods for identifying malware metamorphosis had a success rate of between 65 and 80% (Virvilis, Gritzalis and Apostolopoulos 2013, Fraley 2017).

Other malware related vulnerabilities exist and can be viewed on the Common vulnerabilities and exposure database or the National Vulnerability database (Byers, Waltermire and Turner 2020, Mitre Corporation 2021).

In the modern Internet environment, HTTPS is the essential cryptographic protocol for protecting data integrity and privacy while it is being transmitted between two parties. Different user categories are involved with HTTPS, such as end users who are obliged to make important security decisions when confronted with warnings or administrators who must deal with the principles of cryptography and difficult compatibility issues (Krombholz et al. 2019). HTTPS uses the Transport Layer Security (TLS) or Secure Sockets Layer (SSL) for establishing encrypted communication between servers and clients (Mai et al. 2022).

2.1.4 Transport Layer Security (TLS)/ Secure Socket Layer (SSL)

HTTPS relies on two technologies to provide secure transmission of data, Transport Layer Security (TLS) and/or Secure Socket Layer (SSL). SSL is a security technology that aids in establishing secure communication links between device and sends all data as encrypted messages. This technology is used to secure HTTP communication links which were original transmitted in plain text. The use of SSL technology can be noticed from our browsers with the padlock sign visible alongside the Uniform Resource Locator (URL). This approach uses certificates to authenticate the server. (Duddu et al. 2020)

The Transport Layer Security (TLS) is an improvement on SSL a popular security protocol with multiple versions. TLS has seen various versions including TLS 1.0, TLS 1.1, TLS 1.2 and is currently on the latest version TLS 1.3. This protocol operates under the application layer of the Open Systems Interconnection (OSI) model (Alashwali, Szalachowski and Martin 2019).

Both the SSL and TLS protocol operate using a handshake approach to establish a secure session. The handshake for SSL is presented below.

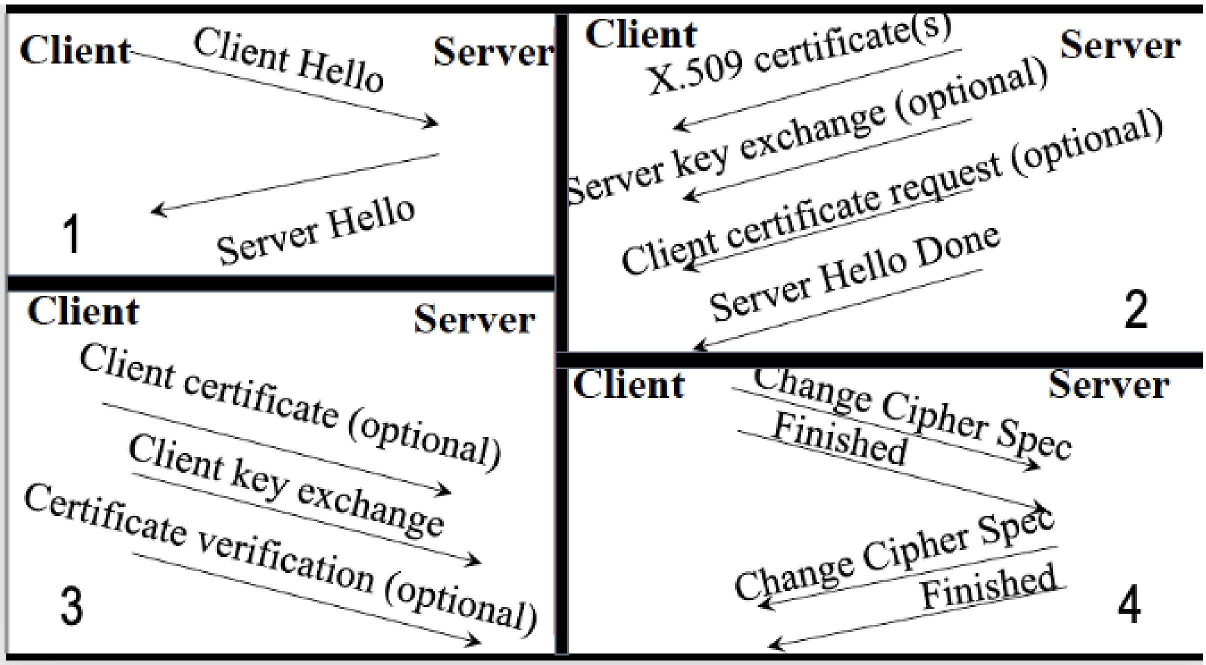

Figure 2.1 4-way SSL Handshake (Devi et al. 2020)

The four stages majorly involve establishing the requirements for both the client and server. The client the authenticates the server and key exchange is done. At this stage it is optional for authenticate the client but the client key is exchanged. Finally change cipher specification protocols by both the client and the server and the session is then established. (Devi et al. 2020)

2.1.5 HTTPS Malware

HTTP a highly used internet protocol needed an upgrade due to the security concerns with shared information. Due to the high number of traffic malwares exploit transmission using this protocol as blocking HTTP traffic became an overambitious move. Encrypting HTTP traffic became a solution using Transport Layer Security (TLS). This protocol provides a secure communication channel between communicating devices (Kohout et al. 2018).

HTTPS has been seen as a good method for preventing HTTP analysis. The security successes in the utilization of HTTPS has caused a shift from HTTP to HTTPs with over 40% utilization by well-known websites. When using HTTPS, a sniffer can only see basic traffic header details (Prasse et al. 2017).

According to Singh (2022), there has been a 50% increase in Network based malware attacks. In recent times malwares utilize SSL/TLS to evade detection even when advanced techniques like deep packet inspection and the traditional signature-based techniques are applied. This approach was majorly used by users to ensure secure communications although this has also equipped adversaries a tool for malicious undetectable transmission. (Singh 2022)

Due to the attention being received from security experts and the increased utilization of HTTPS channels to hide malicious content and behaviour and evade firewalls and intrusion detection devices, a large number of researchers have proposed the use of HTTPs malware traffic classification through machine learning and deep learning techniques. (Ahmed et al. 2022)

The next section presents existing mitigation techniques against HTTPS malware traffic attacks.

2.1.6 Mitigating HTTPS Malware Traffic Attack

Authors have proposed the use various methods with the most popular being classification and other machine learning approaches. Classification approaches have been based on ports, payload, behaviour amongst others (Bader et al. 2022).

Fu et al. (2022) proposed the use of graph-based network analysis in detecting encrypted malware traffic detection. Their work highlighted the dangers associated with this form of attack as the attack is usually accompanied by remote access and possibly a command and control server. They also identified the limitation while dealing with encrypted traffic as vital information that could aid in easy detection are not available. Fu et al. (2022) approach relies on temporal and spatial characteristics of network behaviours.

Deri, Cardigliano (2022) in their work leveraged on cyber score which aids in identifying important cybersecurity events in a network. Their approach was carried out with an open source Deep Packet Inspection (DPI) and network traffic monitoring tool. Numerical values are then attached to measure the relevance of the current activity to cybersecurity threats.

Bader et al. (2022) Proposed the use of MalDIST classifier on PCAP files, their approach evaluated the success rate of Support Vector Machine (SVM), Random Forest (RF) and K-Nearest Neighbour, together with deep learning algorithms.

Han et al. (2022) proposed the use of a lightweight unsupervised anomaly detection approach on encrypted Malware Traffic. Han et al. (2022) used a three-layer autoencoder on encrypted data as HTTPS traffic is transmitted encrypted form source to destination.

These are a few recent mitigation methods against malware HTTPS traffic. The foregoing section presents related works in the mitigation of HTTPS malware traffic using machine learning approaches.

2.2 Related Work

A number of researchers have worked tirelessly to develop solutions to the insecurity associated with information systems. Security related challenges have gone advanced as more attack sophisticated attack methods surface. HTTPS malware attacks have been in existence for over a decade and still persists in attack successes (Anderson and McGrew 2017). Various approaches and methods for preventing and mitigating this threat have been reviewed and analyse. As shown in Table 2.3, approaches and methods are highlighted in a comparative evaluation table. Table 2.3 is subdivided to provide information on the author and year of publication, title, outcome, and limitations.

Table 2.3: Comparative Evaluation Table of Related Work

S/n	Author/ Year	Title	Outcome	Limitations
1	Wang et al. (2017)	Malware traffic classification using convolutional neural network for representation learning	Most effective traffic representation is session. Attained an average accuracy of 99.41% using data from 20 sources including Facetime, Skype, Gmail, outlook amongst others.	The results of the research show great accuracy in classifying malware traffics. This work deals with unencrypted malware traffic and it is also using targeted malware datasets to 20 known web applications thereby reducing its scope
2	Marin, Caasas and Capdehourat (2021)	Deepmal-deep learning models for malware traffic detection and classification	The results of the underlying statistics of the datasets showed a difference in payload sizes of normal vs malicious. Raw packets detection using DeepMal outperformed Raw Flows with and AUC of 0.998 vs an AUC of 0.928 for Raw flows. The model achieved an accuracy of 77.6%	The results from the use of DeepMal is impressive even as the issue of handcrafted features was eliminated and high accuracies were attained. The limitation of this work is that this solution applies to unencrypted or HTTP traffic which is gradually being faced out by organizations.
3	Anderson and McGrew (2016)	Identifying encrypted malware traffic with contextual flow data	The results show an improvement in accuracy when multiple traffic data types are utilized. This include HTTP, TLS, DNS, SPLT and BD.	The approach showed great signs in detecting malware traffic through the combination of data for different protocols. This approach has a heavy utilization of

			This achieved an accuracy of 99.97% using l1-logistic regression classification. Using a 10-fold paired T test, no significant improvement was identified with a 5% SI.	resources and evaluates traffic in blocks as other data must be collected for detection. Not just an instance of data traffic.
4	Bovenzi, Cerasoulo and Montieri (2022)	A comparison of machine and deep learning models for detection and classification of android malware traffic	Using Precision, recall and F-measure as the evaluation metrics Random Forest outperformed the Decision tree and 1D-CNN with a recall of 97 and F-measure of 86 while using a flat approach and a precision of 97 while using hierarchical. The closest to this was 1D-CNN with a precision of 97 while using flat.	This research work evaluates the effectiveness of machine learning and deep learning approaches to detecting and mitigating android malware traffic. The results shows machine learning approach using random forest is more effective. This work is limited by its focus on just android services and concentration on unencrypted traffic.
5	Shire et al. (2019)	Malware Squid: a Novel IOT Malware Traffic Analysis Framework Using Convolutional Neural Network	The approach yielded an accuracy level of between 60% and 91%. This improvement with the accuracy was with every test. There was	The results of this work using CNN showed great results in effectively detecting traffic malware. This approach also learns from previous misclassification and

		and Binary Visualisation	also a steady decline in false positive rates from 40% to 5%. The DDoS detection showed the best accuracy which only presented 16% of the entire dataset. The final test achieved a 91.32% accuracy with precision, recall and F1 value as 91.67%, 91.03% and 91.35% respectively.	improves in every subsequent test. This approach is limited by its focus on unencrypted data and also the heavy computing resources in converting the traffic data to 2D images.
6	Wang, Fok and Thing (2022)	Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study	From the 10 algorithms used with the 5 datasets separately Random Forest had the 3 top accuracies out of 5. With 92.07%, 94% and 94.71%. The research showed that the use of multiple datasets for encrypted traffic is more effective.	This work shows high accuracies in data malware traffic even with encrypted data to detect concealed malware traffic. This method is effective in detecting HTTPs malware traffic. The only limitation is that due to the sensitivity of cyber security issues the accuracy needs to be improved to reduce the occurrence of false positives.
7	Lichy et al. (2023)	When a RF beats a CNN and GRU, together—A comparison of	The results from the research showed that Random Forest outperformed the other	This research show commendable results in detecting malware traffic using combined datasets.

		deep learning and classical machine learning approaches for encrypted malware traffic classification	machine learning algorithms while, MalDist outperformed the other deep learning algorithms. When comparing between machine learning and deep learning, random forest outperformed all the deep learning algorithms. Among the three datasets MTAB had the highest accuracy. Results also show high accuracy in detecting zero day malware	The combination of both encrypted and unencrypted data might affect the models accuracy when applied to real life network with the transmission of encrypted data.
8	Tang et al. (2020)	Malware Traffic Classification Based on Recurrence Quantification Analysis	The dataset used in the dataset are 10 including HTTPS, P2P, Zeus, SFPT, Hangouts, Trickbot, Sennoma, SMB, Artemis and Miuref. The accuracy with under sampling was higher than the accuracy when under sampling is combined with SMOTE. Results also show that the length between 60	This approach achieves a 96.78 accuracy in adequately classifying a multi-vector dataset. The results show that the number of features is not positively correlated to the accuracy and that the accuracy peaks at 80 features.

			and 80 had the highest accuracy.	
9	Letteri et al. (2020)	MTA-KDD'19: A Dataset for Malware Traffic Detection	Features with high correlation >95 were removed, duplicates were removed, zero, unavailable and missing values were removed. Multi-layer perceptron was used and achieved an accuracy of 99.69%	This research work shows high accuracy in detecting malware traffic. This approach applied neural network to a malware traffic dataset. The dataset used consists of unencrypted data inline with HTTP traffic but ineffective against HTTPs traffic.
10	Celik et al. (2015)	Malware traffic detection using tamper resistant features	The results show that across malware families there is a high occurrence of code reuse. The results also show that recent attack vectors had more false positives and false negative rate showing that recent attack malwares are more evasive in traffic detection. Using AUC to evaluate the models K-NN outperformed OCSVM, LSAD and K-means.	The results of the work show prominent results in distinguishing between traffic from legitimate device and traffic from infected device. This research is limited to HTTP traffic from devices and might be ineffective when handling encrypted traffic used be most applications and malwares. The results from the model needs more evaluation metrics to evaluate the model’s effectiveness.

11	Isingizwe et al. (2021)	Analyzing Learning-based Encrypted Malware Traffic Classification with AutoML	While considering the feature importance incoming bytes was the highest. The classification for benign showed high precision and recall. The tool Cisco Joy provided an effective way of collecting encrypted traffic data.	The results show the effect of using multiple algorithms in encrypted malware traffic detection which provides high accuracies with low false positive rates. The limitation of this solution is the use heavy computing resources especially for hyper parameter tuning. The work did not also state the accuracy of the proposed model and only showed prove of concept.
12	Barut et al. (2022)	R1dit: Privacy-preserving malware traffic classification with attention-based neural networks	The results of this work outperformed the state of the art with an F1-score of 0.639 against 0.636. Using self-attention-based deep learning model with an accuracy of 67.9%. The final results showed an F1-score of 0.975. Residual 1_D Image Transformer (R1DIT) model also achieves a 99.99% accuracy in	The results show that malware traffic detection using R1DIT show efficiency while preserving privacy. Dataset included both encrypted and unencrypted data. The high accuracy detected was based on unencrypted data and therefore its performance on HTTPS malware traffic is not confirmed.

			detecting TLS1.3 DDoS traffic.	

Table 2.3 presents twelve related works on machine learning and anomaly-based detection approaches to detecting malware traffic both encrypted and unencrypted traffic. This table was built from existing articles using the keywords; malware traffic detection, Machine learning detection of malware traffic, HTTPS malware traffic detection and encrypted malware traffic detection.

The major problem identified is the inability to protect devices and organizations from encrypted malware traffic devices (Lichy et al. 2023, Wang, Fok and Thing 2022, Shire et al. 2019, Bovenzi et al. 2022, Marín, Caasas and Capdehourat 2021, Wang, Zeng and Sheng 2017). The others identified challenges in preservation of privacy which decrypting and analysing traffic as most communication channels utilize encrypted traffic to maintain privacy, and ineffectiveness in detecting encrypted malware traffic using port-based and signature-based detection.

The approaches in mitigating malware traffic as seen on Table 2.3 utilized machine learning and deep learning approaches. These approaches provided various sets of accuracies based on various datasets. Five authors utilized multiple datasets to train their models while the others used a single dataset.

Among the approaches using machine learning and deep learning, the most popular algorithms used were Convolutional Neural Network (CNN) and Random Forest (RF). The CNN approach utilized both 2D and 1D.

Wang et al. (2017), Marin, Caasas and Capdehourat (2021), Bovenzi et al. (2022), Shire et al. (2019) and Lichy et al. (2023) used CNN in in building a malware traffic detection model. While Shire et al. (2019) used 2D-CNN using TensorFlow for the detection model 100 instances of data distributed as 30% normal and 70% malicious through 500 iterations. Their approach achieved a final accuracy of 91.32%. Shire et al. (2019) work presented accuracy for each attack vector limiting its effectiveness in detecting unknown attack vectors. Their approach used a single method in detecting traffic malware. Others combined the use of CNN with other Deep learning and machine learning algorithms. Wang et al. (2017) combined CNN with static rule based approach achieving an accuracy of 99.41%. Their work focused on detecting HTTP malware traffic thereby limiting the scope of traffic detection to unencrypted data.

Marin, Caasa and Capdehourat (2021) integrated LTSM to CNN in detecting malware traffic. The LTSM formed a major part of the recursive internal layers. Their wok achieved 77.6% accuracy in detecting HTTP malware traffic. Lichy et al. (2023) and Bovenzi et al. (2022) both used CNN and Random Forest, for the two researches Random Forest outperformed CNN, making RF a more effected algorithm for detecting malware traffic.

The other approaches used machine learning algorithms majorly Random forest which showed the most effective results in detecting malware traffic.

2.3 Machine Learning Definition and Application

In his first description, Arthur Samuel describes a computer as having the capacity to learn without being explicitly told what to do. Tom Mitchell changed this meaning to a more appropriate one. He described machine learning as a system that has enhanced its capabilities as a result of experience (Kim, G., Choi and Choi 2018). On the basis of information learned in the past, the machine can respond to events.

As a consequence of utilising machine learning in expert system development, ground-breaking machines that can identify narrowly focused diagnostic issues have been created (Rajkomar, Dean and Kohane 2019). With the big data industry expanding quickly, the health sector in particular has achieved outstanding outcomes in the early disease detection. With a forecast accuracy of 94.8%, Chen et al. (2017) were able to identify regional chronic disease of cerebral infarction using a convolutional neural network (CNN) based unimodal risk prediction algorithm. Due to this, there are now fewer possibilities of an outbreak and an early diagnosis of the disease. Machine learning has been used to address a huge number of health-related issues, most of which used early detection methods to save lives.

The application of machine learning in biology has grown to include, among other things, the categorization of signal peptides, pupylation site prediction, somatic mutation profiles, glioblastoma, and microscopy. (Arganda-Carreras et al. 2017, Ehsan et al. 2018, Silva et al. 2019)

Since detection and prevention of security breaches have their limitations, a significant application of machine learning methods is in the field of information security. This is used for firewalls, network traffic analysis, spam message detection, and encryption, among other things. (Sharma et al. 2017, Moon et al. 2014)

2.3.1 Machine Learning Algorithms

Without human intervention, machine learning is a technique for reacting to occurrences based on experience. Different methods known as algorithms are used to execute this problem-solving strategy. This can be divided into two categories further: supervised and unsupervised learning.

2.3.1.1 Supervised Learning

A user-involved learning process using a labelled training collection is referred to as supervised learning. In supervised learning, a label is given to each occurrence of data in the dataset based on the name of the class. The dataset contains both the intended output and the anticipated input object. The dataset is split in half, typically in the ratio of 80:20, in order to assess the precision of the data classification. The training dataset is made up of 80% of the data, and the testing dataset is made up of 20% of the data. The model is taught using the training data, which makes up 80% of the information. The instruction set is where all of the information is found. The method to be employed is determined by algorithms. (Weitschek, Fiscon and Felici 2014, Harrington 2012)

2.3.1.2 Unsupervised Learning

This strategy utilises data without labels, categories, or classes in order to learn more. Unsupervised learning is a technique that becomes more effective as it encounters more events, so the model's precision will progressively increase as it learns more. Cluster-based algorithms and dimensionality reduction are the two main algorithms connected to unsupervised learning. (Celebi and Aydin 2016, Finn, Goodfellow and Levine 2016)

The detailed discussion of malware traffic detection in this chapter emphasises their current vulnerabilities and available mitigation strategies. To accurately define the issue, a comparative and systematic review approach was done. The comparative and systematic study also demonstrated how current approaches affect HTTP and HTTPS malware traffic detection, prevention, and mitigation.

2.4 Machine Learning Tools

There are many tools used in machine learning for creating models. A few of the most popular machine learning tools are discussed below:

1. TensorFlow: Developed by Google, TensorFlow is an open-source machine learning library used for various applications in deep learning.
2. Keras: Keras is a high-level API for building and training deep learning models that sit on top of TensorFlow as an interface.
3. PyTorch: PyTorch is an open-source machine learning library known for its flexibility, ease of use, and dynamic computation.
4. Scikit-learn: Scikit-learn is an open-source Python library for machine learning. It is simple and efficient for data mining and data analysis tasks.
5. Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays.
6. Caffe: Caffe is a deep learning framework that allows you to train and test neural networks with speed, accuracy, and scalability.
7. MXNet: MXNet is a flexible and efficient deep learning library that allows you to train and deploy deep learning models on various devices.
8. Caret: This is an open source library for r language that allows you to train various machine learning algorithms and present summarized results.

(Singh, Singh & Sarkar, 2022, Niraula, El Naqa, 2022)

These are some of the most popular and widely used ML tools by researchers and data scientists today. Each tool has its unique features and benefits that make it suitable for different types of ML applications

Both Python and R are popular languages for machine learning, and both have their pros and cons. Ultimately, the choice of which language to use depends on the specific needs and preferences of the user. However, Python is usually preferred by most due to its more extensive libraries, community, and support for deep learning. Nonetheless, R's advantage is its statistical capabilities that make it an excellent choice for data manipulation, visualization, and data science tasks. (Dangeti 2017)

This project will use R dues to its capabilities in data manipulation and visualization and my experience and expertise in R language.

The next chapter presents the project specification for this project using machine learning.

CHAPTER 3: PROJECT SPECIFICATION

3.1 Aims and Objectives

This project aims to improve the current detecting techniques that have already been put in place.

i. To build a machine learning model for detecting HTTPS malware traffic without having to decrypt.
ii. To validate the model and improve its performance through training and testing.
iii. To evaluate the model performance with other algorithms and assess the efficiency of the proposed system.
iv. To document the system's testing, evaluation results and their applicability to the scientific repository of cybersecurity knowledge.

3.2 Functional and Non-functional Requirements

The functional and non-functional requirements are presented in section 3.2.1 and 3.2.2.

3.2.1 Functional Requirements

The functional requirements are presents using the MoSCoW prioritization approach. This provides grouping the requirements based on importance thereby meeting up with critical requirements before others. The categorization is Must have, Should have, Could have and Will not have. This is presented using a tabular approach below.

Table 3.1: Functional Requirements Using MoSCoW Approach

S/N	REQUIREMENTS	MoSCoW
1	Four machine learning algorithms will be evaluated to select algorithm with the highest accuracy score.	Must
2	Solution will analyze large amounts of data to identify patterns and make predictions	Should
3	Solution will have the ability to analyze encrypted traffic	Must
4	Evaluation of the predicted classes must be confirmed using two evaluation metrics including accuracy and confusion matrix	Should
5	Integration with visualization and reporting tools.	Could
6	Solution must present over 90% accuracy in detecting malware traffic	Must

7	Recall, Precision and F-Measure should be used to confirm predicted classes	Should
8	Ability to handle missing or incomplete data	Could
9	Integrate solution with physical devices and sensors	Won't
10	Ability to perform automated data preprocessing and feature engineering	Must
11	Implementation of machine learning solution on Security information and event management (SIEM) systems	Won't
12	Put machine learning model into production through built API	Could
13	Deployment of solution as standalone desktop application.	Won't

3.2.2 Non-functional Requirements

1. Reliability: The system should perform accurately and should be consistent this will be achieved through the training and testing evaluation results.
2. Fault Tolerance: The system should function as predicted irrespective of unexpected events. This will be achieved by raining the model with a robust dataset.
3. Security: The system will provide security to devices and network traffic. Confidential traffic will maintain its confidentiality as the proposed solution will not utilize deep packet inspection.
4. Scalability: The proposed system will be adaptable to growth as the model will gain experience from robust datasets applicable to any network irrespective of size.

3.3 Methodology

This chapter describes procedures to be utilized in investigating, finding a solution to the research problem, and providing a detailed approach plan. The detailed plan and technical approach provided are about the inference, accuracy, and relevance for the recommendation and application of techniques. This is in identifying, collecting, and analysing information and data used in mastery and comprehension of the research problem. Hence, proving that the outcome of the research work is reliable, valid, and reproducible. We are applying a waterfall model approach to meeting with the aim and objectives of this project.

The first part identifying research goals, has already been achieved in the previous chapter with the objectives and comparative evaluation review of related works. The data selection stage provides data samples for the machine learning stage. The machine learning group stage consists of four recursive stages that performs the learning operation using four algorithms in an attempt to select the best, based on accuracy and F-measure. The output of this is feed to the development of an enhanced prediction model. Figure 3.1 shows the workflow for the machine learning model.

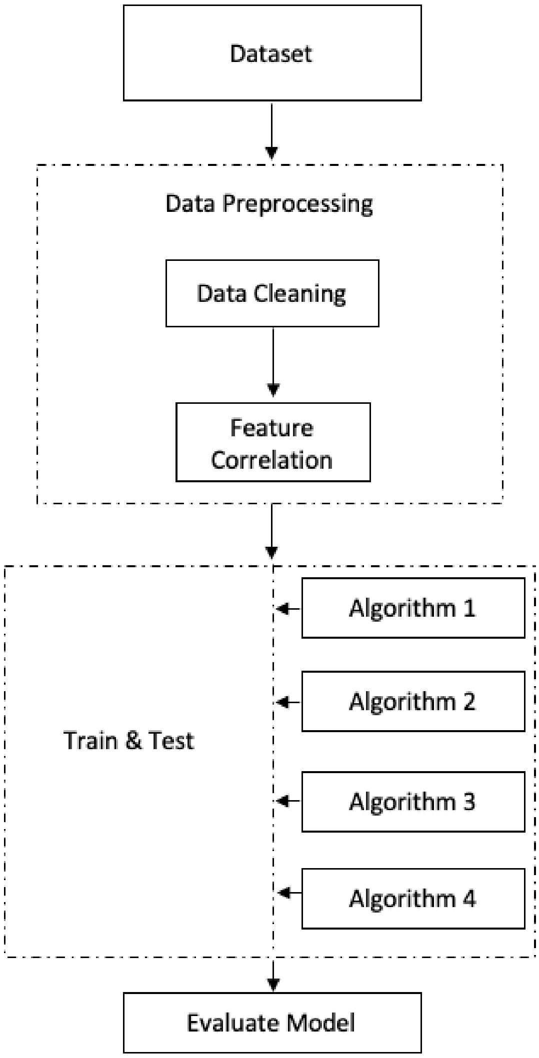

Figure 3.1 Workflow Diagram for Machine Learning Approach

The selection of the datasets to be used in the machine learning exercise is the first step. These datasets are collections of instances of encrypted network traffic.

3.3.1 Dataset Selection

The dataset that will be used has been carefully chosen to contain variables that can be readily gathered from a packet header and are useful for categorizing data. The dataset presents instances of encrypted data holding benign, Bot attack and DOS attack traffic. The dataset is made up of 887,993 malicious traffic and 1,209,156 benign traffic. The Dataset is sourced majorly from Github under the Stratosphere Lab. The dataset is called Intrusion Detection Evaluation Dataset CICIDS 2018. These datasets are PCAP (Packet Capture) files of internal network traffic. (Sharafaldin, Lashkari and Ghorbani 2018). The CICIDS dataset did not meet the requirements of the proposed solution and did not include some very important features including source port. This led to the utilization of another published fully encrypted traffic dataset called CIRA-CIC-DoHBrw-2020 (MontazeriShatoori et al. 2020). The CIRA-CIC-DoHBrw-2020 dataset was sourced from UNB. The dataset also consists of PCAP files containing benign encrypted traffic and Malicious encrypted traffic. The malicious traffic file contains 19,807 instances of data while the benign traffic file consists of 249,836 instances of data. The dataset also presents 34 features with a label making 35 columns.

3.3.2 Data Cleaning

Datasets are frequently dirty when they are gathered. Datasets that comprise one or more of the following are considered dirty datasets: values that are unexpected, irrelevant, the wrong data format for the data, unexpected duplicates, and inconsistency. Scaling, normalization, removing the complete row, or replacing missing values with either the mean or median value for the column are techniques used to clean the data. Clean data increases prediction model accuracy because inaccurate data may be deceptive and thus impact the output. Cleaning can be done directly or automatically using programs like the Python scripting language, the Waikato Environment for Knowledge Analysis (WEKA), R, or Anaconda. R language will be used for this project.

R also provides methods for examining datasets to improve the model's accuracy. Methods available are examining column datatypes, missing (NaN) values, uniformity and consistency.

3.3.3 Feature Selection

The next step is to select the variables or features from the collection that are of interest in building a model with high accuracy. These variables are chosen because they can help detect the existence of malicious traffic or applications in HTTPs traffic. The system and classifier will operate more quickly if the variables are reduced; the more variables, the longer it will take to handle them. These features will be chosen using three different techniques. chi^{2}test-based univariate feature selection for positive prospects See equation 3.1, feature importance to show features likely to influence accuracy and by percentage illustrated with matplotlib, pyplot, and ExtraTreeClassifier, as well as correlation matrix to demonstrate positive correlation between features using a coloured graph matrix. The datasets that will be used have a large number of features, and by reducing these features to just the ones that matter most, the predictive modelling algorithm will perform more accurately. This is basically selecting a subset of features present in dataset. If features that are filtered out are still present, the predictive model's efficiency and precision may both suffer. The process of feature selection is a win-win strategy that, in addition to raising accuracy, reduces the amount of data that needs to be analysed, speeding up the development of the predictive model.

$$X^2 = \frac{N(AN-MP)^2}{PM(N-P)(N-M)} \tag{3.1}$$

Where:

N = total number of instances

M= the number of instances that feature X

A= the total number of positive instances that contain feature X

P = the total number of positive instances

With the relationship M = A+B, P=A+C, N-M =C+D and B+D=N-P

The given dataset served as the source of the variables for the calculations, which can be applied to datasets of different sizes. As shown in Figure 3.2, the Feature importance offers a graph that displays the results in bars along with their relative levels of importance.

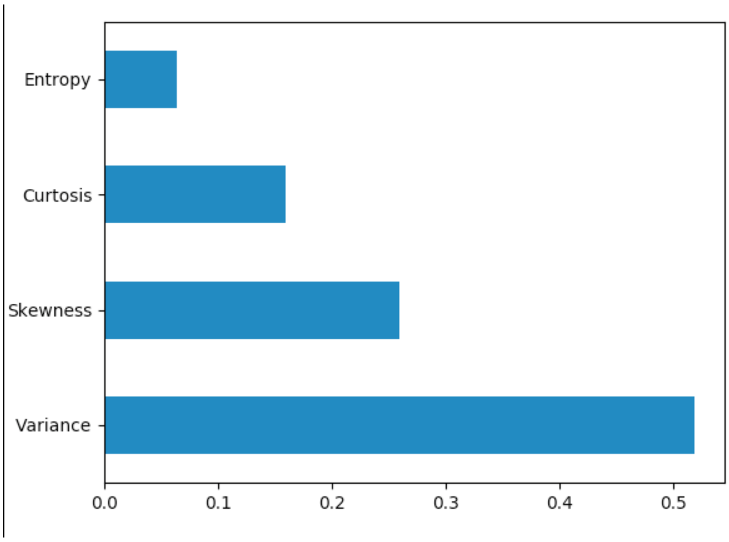

Figure 3.2 Feature Importance Result Sample

Using a correlation matrix to look for relationships between the characteristics is the last step in the selection process. This approach makes use of correlation analyses and displays the outcomes as an X by X matrix. The correlation coefficient's result ranges from -1 to +1. -1 indicated by red and +1 as green. Additionally, it measures the degree and orientation of association. The stronger the association, the greater the value or the nearer +1. As seen in Figure 3.3.

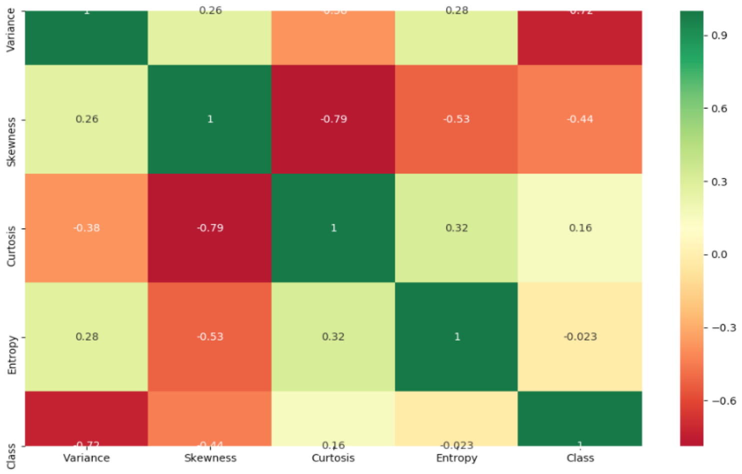

Figure 3.3 Correlation Matrix Sample

3.3.4 Split Dataset

There are two methods for performing anomaly detection using machine learning algorithms. While the other uses supervised methods, one uses unsupervised techniques. When there is no pattern or label in the existing dataset, unsupervised methods are used, whereas the supervised method is the opposite. The dataset to be used with supervised techniques is usually split in two halves, the first half is used to train the machine on normal versus malicious traffic, while the second half to test the accuracy of the algorithm in detecting and efficiently classifying every instance of data presented to the machine. A R tool will be used to divide the dataset using an 80:20 ratio, 80% for training and 20% for testing. Providing data for both training and testing using the formula in equation 3.2.

$$\text{Accuracy} = \frac{a}{b} \tag{3.2}$$

a = Number of correct predictions

b = Total Number of instances used to test the system.

3.3.5 Training Model

Once the data has been gathered and labelled, it can be used to distinguish between malicious and legitimate network activity. At this stage, the classifier, which is another name for the algorithm used to classify data, is chosen. The correct insertion of the data into the dataset is known as categorization of the data.

The dataset provides the system with a knowledge base to help the system learn about the events and also how to properly classify them, which is necessary for a machine to respond and react to events appropriately. Since labelled data have predefined patterns, this method is typically used with that data.

The algorithms chosen were done after their use in analogous situations. K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Decision Tree, and Random Forest are some of the methods in this group.

The result of interest is the accuracy level of each algorithm. Also used for evaluation are recall, precision and F-measure.

1. K-Nearest Neighbor: As shown in Figure 3.4, this algorithm simply groups data points according to their distance and the K value given.

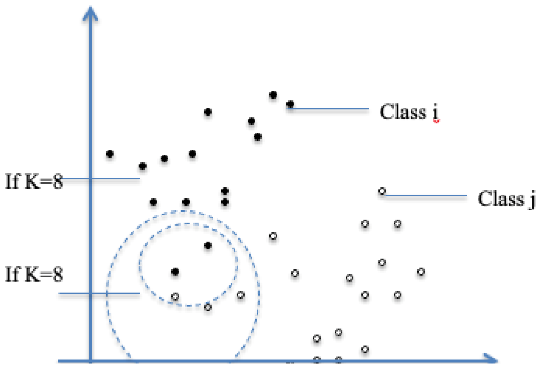

Figure 3.4 Graph showing relationship between i and j for KNN

K is a constant used to determine the number of points per group. The value of K influences the training results. The Euclidean distance, as shown in equation 3.3, is the distance used in choosing data points for clustering.

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2)} \tag{3.3}$$

2. Support Vector Machine: To create the support vectors that are used to characterize the planes, this algorithm creates data from the datasets. Equation 3.4 illustrates the definition of the training group for SVM. Equation 3.5 shows the formula used to produce the separation of hyper plane used in SVM. Equation 3.6, a quadratic programming problem, is solved to yield the hyperplane. Figure 3.5 displays an example of the results from equation 3.6.

$$X = \{x_i, y_i\}_{i=1}^{n} \tag{3.4}$$

$$w^T \varphi(x_i) + b \tag{3.5}$$

$$\min_{w,b} j(w) \frac{1}{2} ww^2 + C \sum_{i=1}^{n} \xi_i \tag{3.6}$$

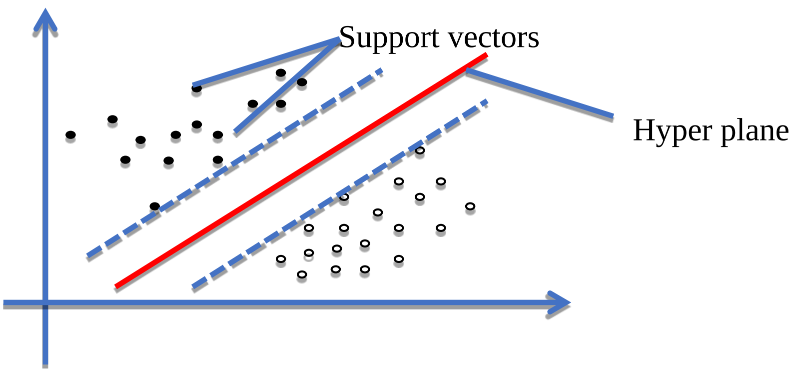

Figure 3.5 Hyper plane example for SVM

3. Decision Tree: When classifying data, a decision tree algorithm employs a structure resembling a tree. Using the labeled dataset, a tree-like structure that expands in reverse sequence is created. The nodes of the tree are broken down into judgment nodes and

terminal nodes known as leaves. Edges that link parent nodes to child nodes are used to join these nodes. A factor is used by the decision tree to stop the tree from growing. With each growing component, the tree grows in a recursive manner. At each judgment node, the nodes are divided to create child nodes. Entropy, categorization error, or Gini are taken into consideration when dividing this data. Probability is used to determine these three criteria.

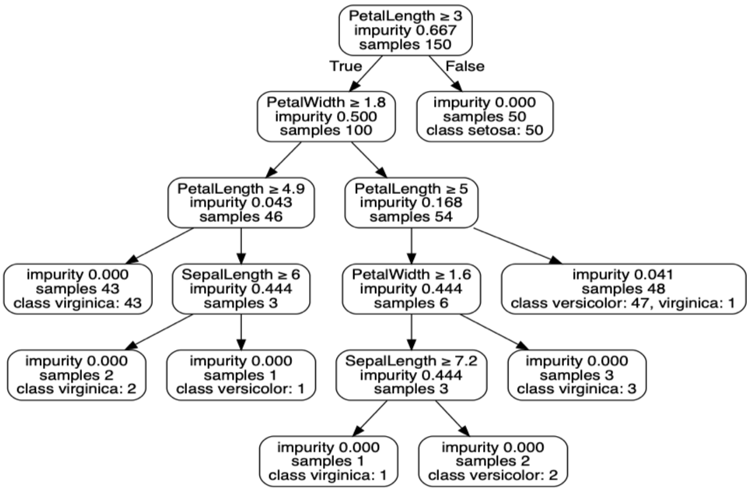

Figure 3.6 Decision Tree Example

A decision tree constructed from iris sample dataset characteristics is shown in Figure 3.6. At each decision node, the tree's branching process proceeds. The leaf is reached when this recursive procedure reaches leaf based on the impurity score. The data is categorized as Virginia or versicolor at each leaf based on the dataset used.

4. Random Forest: When classifying datasets, the Random Forest classifier method employs the decision tree strategy. During the training phase of Random Forest, numerous trees are constructed. The data is classified using multiple trees constructed using the decision tree formula. Each tree's output is then displayed, and the classification process is finished by using a majority vote. Figure 3.7 shows a diagrammatic depiction of the procedure.

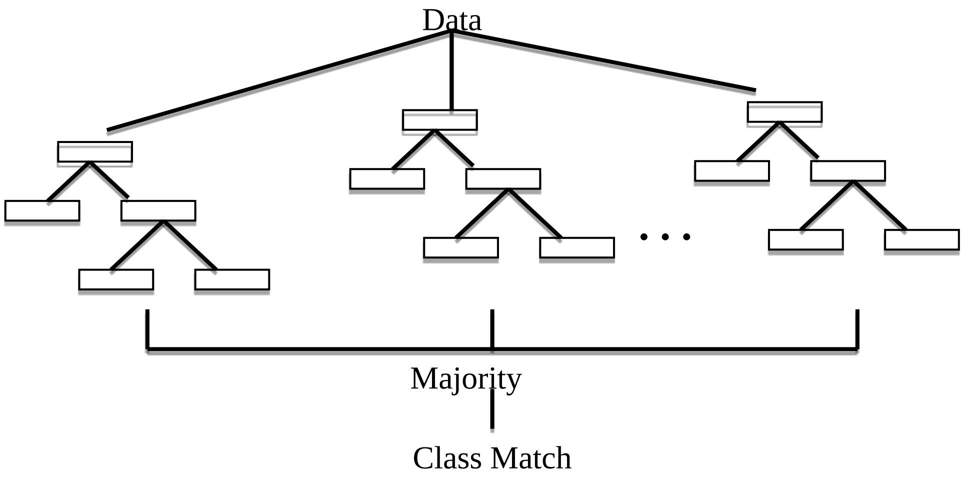

Figure 3.7 Random Forest Example

3.3.6 Testing Model

The machine must be tested to see how effective it is at classifying new instances of data after being trained with half of the dataset's instances. The machine is tested using the second part of the dataset. The accuracy of how well it identifies similar instances can be calculated because every instance in the dataset has labels.

The process must be repeated using four separate algorithms for categorizing data from malware traffic. Table 3.2 shows the results from evaluating selected algorithms.

Table 3.2 Detection result table

S/No	Algorithm Used	Accuracy	Recall	Precision	F-Measure
1	Algorithm 1	In percentage	Between 0-1	Between 0-1	Between 0-1
…	…	…	…	…	…
4	Algorithm n	In percentage	Between 0-1	Between 0-1	Between 0-1

The process of calculating accuracy, recall, precision and F-measure is automated by introducing a function to the code. The method in equation 3.2 is used to calculate accuracy. A feature confusion matrix, which uses an x by x matrix to show the results based on test dataset matches. x is the total amount of classes. We anticipate a 2 by 2 matrix for the two classes this research employs. The format in which the outcome will be presented is shown in Table 3.3.

Table 3.3 Confusion matrix

Normal	Exact match for normal	Mismatch Normal as Malicious
Malicious	Mismatch Malicious as Normal	Exact match for Malicious
	Normal	Malicious

The overall number of instances in the test dataset can be calculated by adding up all the information in the matrix. The confusion matrix connects the anticipated match and the classifier's match. The accuracy level is calculated by dividing the total of the numbers in the "exact match" boxes by the size of the test dataset. The accuracy level in % is obtained by multiplying the level, which ranges from -1 to 1, by 100.

3.4 Review of Legal, Ethical, Social and Environmental Issues

The legal, ethical and social issues surrounding machine learning based solutions are primarily concerned with the dataset source and content rather than the machine learning model. The data to be used will not be decapsulated to see its content thereby affecting the privacy of data being transmitted. The model will be trained and will utilize packet header details only.

Machine learning projects have the capability to transform the way we live, work, and interact with the world. However, as these projects become increasingly complex, they bring with them a range of ethical dilemmas and legal implications, surrounding issues such as data privacy, algorithmic bias, and environmental impact.

Here are some of the ethical, legal, social, and environmental issues that one needs to consider before implementing machine learning into any projects:

1. Bias and Fairness: If machine learning algorithms are taught on biased data, they may reinforce pre-existing biases. These prejudices, which can result in unfair judgements and discrimination, can be founded on racial, gender, religious, or national origin. This

effort removes biases using network traffic data. To eliminate any bias, the affected rows are deleted before the position is modified as part of the data cleaning procedure.

2. Privacy and Security: Large quantities of data are needed for machine learning algorithms, and the growing adoption of these technologies can pose a serious risk to personal privacy and data security. It prompts concerns about the use of private data without peoples' consent. The aforementioned both encrypted and unencrypted traffic's substance are unaffected. All datasets used are fully acknowledged.
3. Economic and Social Impacts: The widespread use of machine learning techniques may result in employment loss and economic inequality. Therefore, it is crucial to consider how such technologies will affect civilization. However, because it offers security for computing users and devices, this initiative will boost the economy.
4. Environmental Impact: Machine learning initiatives can have an adverse impact on the environment because they consume a lot of energy during training and operation. Making AI models more energy and carbon-efficient requires action from machine learning researchers.
5. Intellectual Property: Regarding who owns the algorithms, codes, and trained models, machine learning initiatives can also raise intellectual property concerns. We have completely acknowledged and complied with the data usage guidelines for the project's purposes.

This project does not in any way contravene the data act laws by gaining control over personal and organizational data.

3.5 Code of Practice and Industrial Standards Related to Work

The IEEE standards association through the standard IEEE P2840 specifies frameworks and architectures for machine learning models trained with encrypted historic data. IEEE P2840 specifies security and technical requirements, functional components, workflows and protocols. The IEEE P2840 standard is tagged standard for technical framework and requirements of shared machine learning.

CHAPTER 4: DESIGN ALTERNATIVES AND JUSTIFICATION FOR DESIGN

4.1 Machine Learning Design Alternatives

Machine learning projects have some differences from traditional software development, many of the same principles can be applied to the development of machine learning projects. Some alternative design approaches that can be used for machine learning projects include: Agile, DevOps and Hybrid amongst others.

The project when using agile is divided into smaller units, or sprints, using the iterative and flexible agile technique. The project develops as a result of the development team's cooperative effort and changes in requirements. This strategy would entail segmenting the project into smaller activities, such as feature engineering, data cleansing, and model selection in the context of machine learning projects. Each work would be finished in an iterative cycle, including input and improvements as the project moved along.

Software development (Dev) and operations (Ops) are combined in the DevOps methodology. (Ops). This strategy places a strong emphasis on automation, teamwork, and ongoing development. The DevOps method for machine learning projects include automating the model selection and deployment process as well as integrating machine learning models into the operational systems.

The hybrid methodology incorporates the greatest features of both the Waterfall and Agile methodologies. Projects involving machine learning that have certain needs that must be fixed and others that may be adjustable can employ this method. The strategy entails segmenting the project into stages, some of which are finished in a linear fashion while others are finished iteratively.

4.2 Justification for Waterfall Approach

In a linear and sequential procedure known as a "waterfall approach," each step of the project must be finished before going on to the next. While the waterfall approach may not be the best approach for all machine learning projects, it is suitable for our solution based on the following reasons:

1. Well-defined stages: The project stages are clearly defined in the waterfall technique, and it is simpler to specify and manage the scope of each step. The phases in the development of an intrusion detection system might include describing the issue, gathering data, pre-processing data, engineering features, choosing a model, training, testing, deployment, and maintenance. The project can be managed and tracked more easily since each step can be clearly defined and under control.
2. Fixed requirements: The success of the project may be judged by how successfully the system detects intrusions, and the requirements for intrusion detection systems are frequently well-defined and established. The waterfall methodology enables the defining of the criteria at the project's outset and guarantees that these needs are met.
3. Traceability: The waterfall methodology enables requirements, design, and testing to be traceable. This makes it simpler to track the implementation, testing, and verification of a specific feature or need. For intrusion detection systems, where traceability is essential for auditing and regulatory reasons, this is especially significant.
4. Sequential nature: With regard to intrusion detection systems, the waterfall approach's sequential nature can be advantageous because it enables the discovery and correction of problems before moving on to the next stage at each level. This minimises the likelihood of substantial issues being discovered late in the project by ensuring that problems are detected and remedied early on.
5. Fixed budget and timeline: The waterfall method enable the project's fixed budget and timetable to be established at the outset. This makes it simpler to manage the project within the constraints of the resources at hand and guarantees that it will be completed on schedule.

4.3 Alternative Approaches to Encrypted Traffic Detection

Traditional malware detection methods may face major difficulties when dealing with encrypted software. However, a number of methods, such as the following, can be used to identify malware that is encrypted.

Behaviour-based detection examines the programme's behaviour rather than its source code. Behaviour-based detection can spot any resemblances between encrypted malware's behaviour and that of known malware.

Heuristic detection entails searching for behaviours and patterns that are typical of malware. It can be useful for finding encrypted malware that is meant to elude detection using classic signature-based methods.

Running the programme in a controlled setting while conducting a dynamic analysis will allow you to watch how it behaves. Dynamic analysis can be used to detect certain behaviours that encrypted malware could display.

Signature-based detection can still be used to identify some encrypted malware versions that have not been adequately disguised, even if it is not effective for all forms of encrypted malware.

4.4 Justification for Choice of Machine Learning in Encrypted Traffic Detection

Due to its capacity to identify patterns and abnormalities in huge datasets, including encrypted information, machine learning is being employed more and more in the detection of encrypted traffic. Traditional traffic analysis methods, which rely on scanning unencrypted network packets to find patterns of interest, have a considerable barrier when dealing with encrypted communication. In order to detect risks in encrypted traffic, machine learning-based algorithms have become a potential alternative.

Machine learning's capacity to spot obscure patterns in encrypted communication that could be signs of malicious activity is one of its primary advantages for the detection of encrypted traffic. Large amounts of encrypted traffic data can be analysed by machine learning algorithms to find patterns that may be hard to find or impossible to find using traditional techniques. Machine learning algorithms, for instance, may spot traffic patterns that point to botnet activity, such a lot of connections to command-and-control servers or sudden surges in traffic.

The capability of machine learning to adapt to new data and learn from it is another benefit for detecting encrypted communication. The continual evolution of encrypted traffic patterns may make it difficult for traditional rule-based methods to keep up. On the other hand, machine learning algorithms are better at identifying and reducing dangers in encrypted communication because they can continuously learn from new patterns and threats.

Overall, machine learning is an effective method for detecting encrypted communications since it allows users to spot hidden patterns and adjust to new dangers. The use of machine learning for threat detection is projected to become more crucial as encryption becomes more common in network traffic.

Studies have shown that KNN, SVM, DT, and RF are effective machine learning algorithms for detecting malicious traffic in encrypted network traffic. While the performance of these algorithms may vary depending on the specific dataset and analysis requirements, they are frequently used and have shown strong performance with high accuracies in multiple recent studies (Wang, Fok and Thing 2022, Celik et al. 2015, Lichy et al. 2023).

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Introduction

A machine learning project must be implemented using a number of crucial phases, including data collection and pre-processing, feature engineering, model training, and model evaluation. This chapter presents the implementation and testing.

5.2 System Requirements

The following are the specifications for the system and applications utilised in the analysis and detection:

1. Operating System: Windows 10 64bit
2. Processor: 2.5 GHz Dual-Core Intel Core i5
3. Storage: 500GB SSD
4. Memory: 16 GB 1333 MHz DDR3
5. Graphic card: Intel HD Graphics 4000 1536 MB
6. R for Windows 4.2.3
7. R Studio 2023.03.0 build 386

5.2 Implementation

The implementation of the solution has two phases categorized based on the implementation and results using the first dataset which could not meet up with the project specifications and in particular the functional requirements. The results for the four models trained using the CICIDS2018 dataset are presented in the Figure 5.1.

```
## Accuracy
##      Min.      1st Qu.   Median    Mean      3rd Qu.   Max.      NA's
## DT   0.7527556 0.7602359 0.7644474 0.7638169 0.7662667 0.7731749 0
## KNN  0.7286888 0.7346592 0.7388263 0.7390028 0.7427795 0.7538197 0
## SVM  0.7490579 0.7571477 0.7605900 0.7603831 0.7636177 0.7729988 0
## RF   0.7016482 0.7150538 0.7190509 0.7198805 0.7262664 0.7326531 0
## Kappa
##      Min.      1st Qu.   Median    Mean      3rd Qu.   Max.      NA's
## DT   0.1937313 0.2059985 0.2147247 0.2154369 0.2219679 0.2612129 0
## KNN  0.2371047 0.2568370 0.2637522 0.2643029 0.2708873 0.2924334 0
## SVM  0.2214274 0.2379003 0.2430096 0.2444804 0.2507479 0.2715362 0
## RF   0.2041923 0.2341986 0.2413481 0.2430946 0.2532174 0.2756860 0
```

Figure 5.1: Results from Machine Learning Training using CICIDS2018

The accuracy with the range of 72% to 76% falling below the expected minimum accuracy and with Kappa less than 0.70 showing that the model will fail when deployed in future. On further examination of the dataset it was discovered that some key features identified in related researches were missing from the dataset including source port. The complete results and other observations are presented in appendix 1. As this did not meet the requirements another dataset was used to meet up with the objectives of the project. The foregoing sections presents the implementation process using the CIRA-CIC-DoHBrw-2020 dataset.

5.2.1 Dataset Collection and Pre-processing

The proposed model is trained using a published secondary historic data called the CIRA-CIC-DoHBrw-2020 dataset. The distribution of the dataset based on the classes is presented in table 5.1. The dataset is a binary dataset.

Table 5.1 Descriptive statistics of the CIRA-CIC-DoHBrw-2020

CLASS	FREQUENCY	PERCENTAGE
Benign	19807	7.35%
Malicious	249836	92.65%
TOTAL	**269643**	**100%**

The R studio tool is used for the implementation of the project solution. The first stage deals with loading the data to the R environment as shown in Figure 5.2.

```
```{r}
Benign <- read.csv("Dataset_Project/l2-benign.csv",header = T) #Load benign dataset
Malicious <- read.csv("Dataset_Project/l2-malicious.csv",header = T)
#Load Malicious dataset
```
```

Figure 5.2: Loading of Dataset for Training using 2nd Dataset

After uploading the dataset to R studio, we need to confirm that there are no missing values in any of the two datasets. The number of missing value is displayed using sum() command.

```
[1] 122
[1] 566
```

The missing values identified in the benign as well as the malicious dataset are handled using the cod snippet below to omit missing values. To make sure the dataset is in perfect condition, a check is done for infinite values as presented in Figure 5.3.

```{r}
Encrypted_traffic= Encrypted_traffic[complete.cases(Encrypted_traffic), ]
#filtering dataset to contain only rows with complete data
```

```{r}
Encrypted_traffic[sapply(Encrypted_traffic, simplify = 'matrix', is.infinite)] <- 0
#filter out infinite data rows
```

Figure 5.3: Filtering out Incomplete Rows in Dataset

5.2.2 Feature Selection

The selection of features is key to the success of the model. The correlation matrix Is used to show feature correlation between features in the dataset. The correlation analysis is first calculated using corrplot library and then presented on a heatmap. The colour green represents high correlation and red represents low correlation. As seen in Figure 5.4, each feature has good correlation with one or more features making all the features relevant to the model.

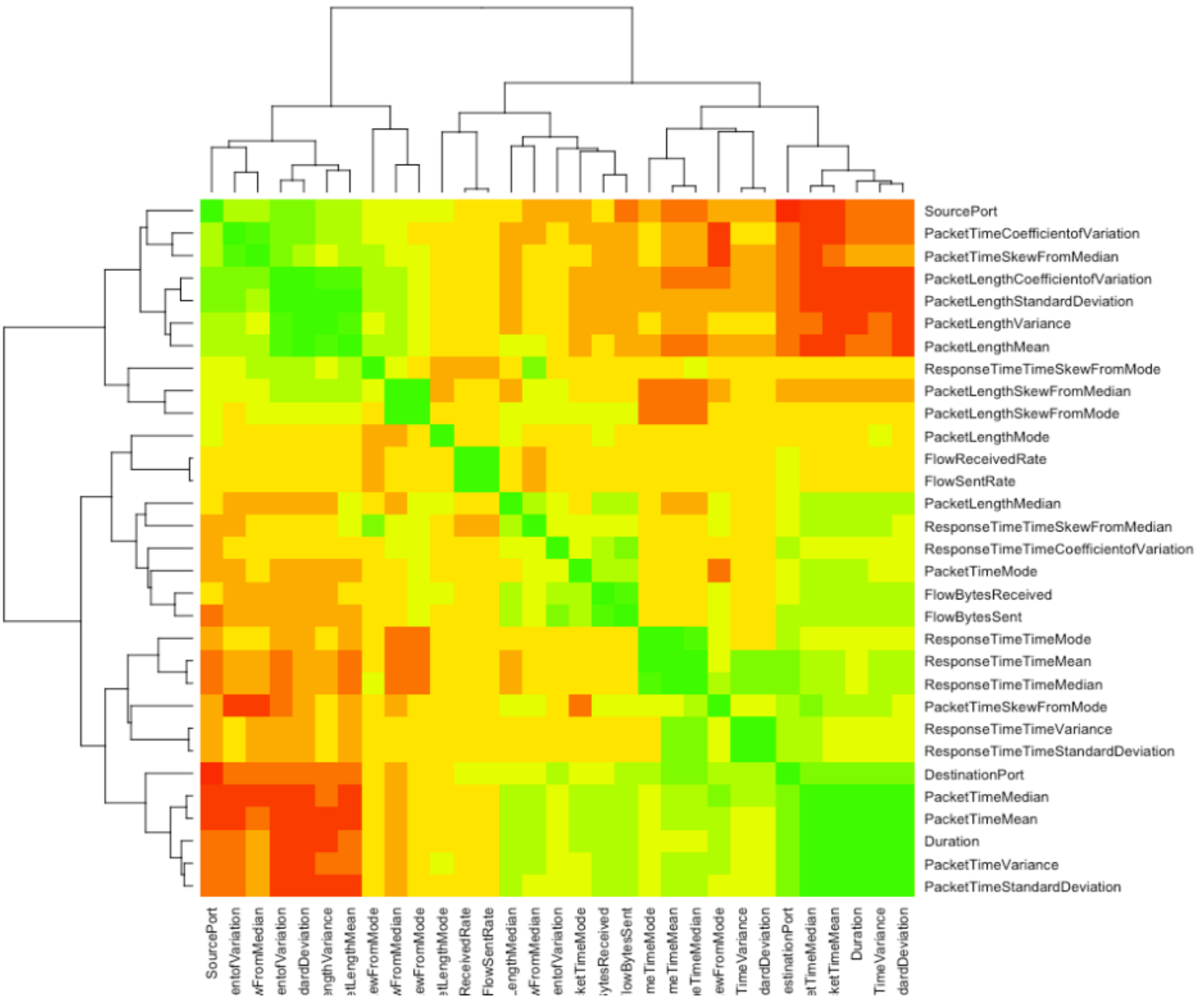

Figure 5.4: Feature Correlation using Heatmaps

5.2.3 Model Training

Before training the model, the class column tagged "Label" is converted from character to factor to aid in the detection process as seen in Figure 5.5.

````
```{r}
Encrypted_traffic$Label <- as.factor(Encrypted_traffic$Label)
#convert label to factor in preparation for model training
```
````

Figure 5.5: Conversion of Label Datatype from Character to Factor

The dataset to be used is then reduced to a subset of the dataset due too system resources and computing capacity as seen in Figure 5.6.

```
```{r}
SampEnc <- Encrypted_traffic[sample(nrow(Encrypted_traffic), 15000), ]
#reduce the number of rows for the training and testing process
```
```

Figure 5.6: Reducing the Size of the Dataset for Effective use of Resources

The Dataset is first divided into two using 80% for training and 20% for testing. The test data is stored as TestDataset and the training data as trainDataset. The ranger and caret are machine learning libraries used for the splitting and model training see Figure 5.7.

```
```{r}
library(ranger) #import ranger for faster random forest model training
library(caret) #import caret for other machine learning algorithms used
index <- createDataPartition(SampEnc $Label, p=0.80, list=FALSE)
#Separate 80% of the data
trainDataset <- SampEnc[index,]
#store the 80% as trainDataset for the training process
testDataset <- SampEnc[-index,]#store remaining 20% as testDatase for testing
```
```

Figure 5.7: Splitting Dataset into Testing and Training Data

The dataset SampEnc is then fed to the four selected machine learning algorithms for training as presented in Figure 5.8.

```
```{r}
fit.cart <- train(Label~., data=trainDataset, method="rpart")
#train decision tree model
fit.knn <- train(Label~., data=trainDataset, method="knn")#train knn model
fit.svm <-train(Label~., data=trainDataset, method = "svmLinear")#train svm model
fit.rf <- train(as.factor(Label)~., data=trainDataset, method = "ranger")
#train random forest model
```
```

Figure 5.8: Training using Decision Tree, KNN, SVM and Random Forest

The results of the four algorithms are presented in Table 5.2 as well as Figure 5.9 and Figure 5.10 using density plot and dot plot with focus on accuracy and kappa.

Table 5.2: Results of the four algorithms

Algorithm	Accuracy	Kappa
Decision Tree	98.12%	90.87%
KNN	97.34%	78.51%
SVM	97.11%	76.59%
Random Forest	99.86%	98.89%

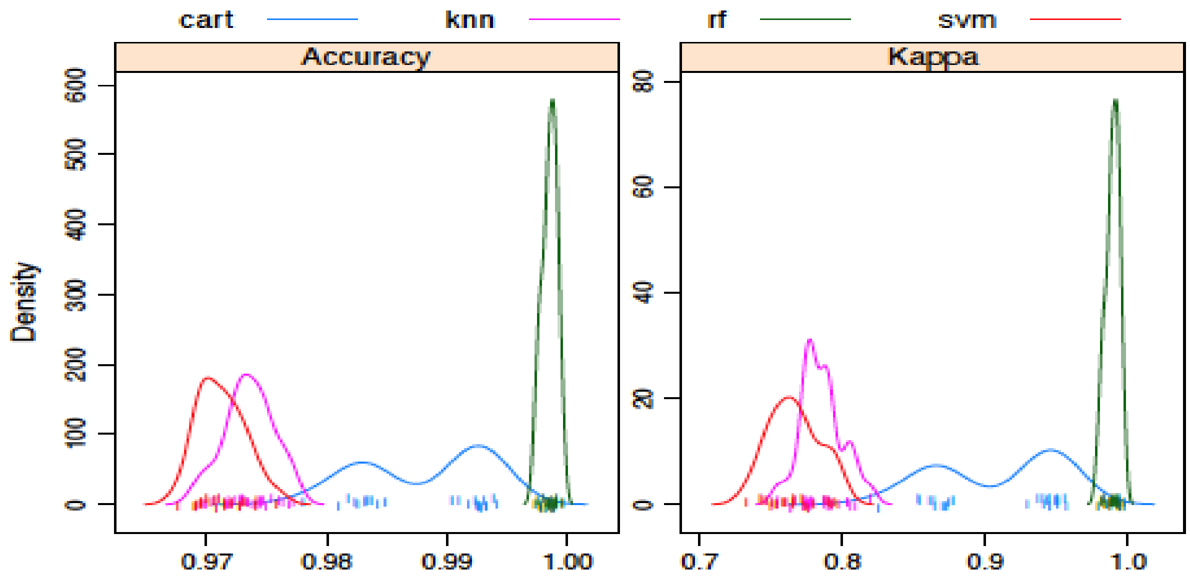

Figure 5.9: Density vs Accuracy for Decision Tree, KNN, SVM and Random Forest

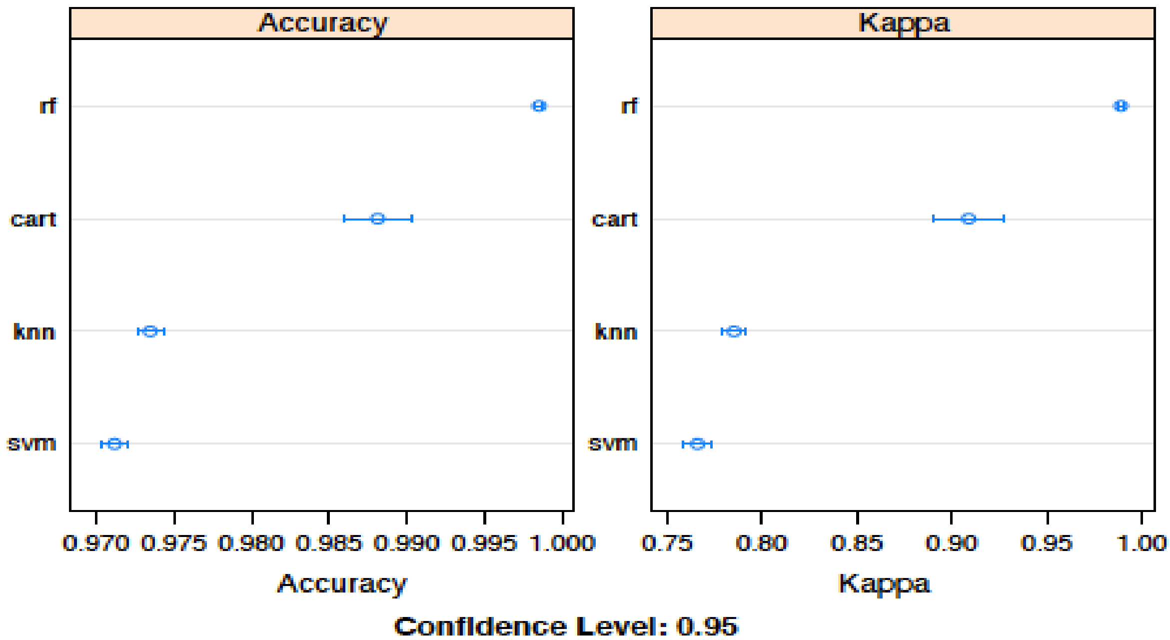

Figure 5.10: Dot plot for Accuracy for Random Forest, Decision Tree, KNN and SVM

From the results seen Random Forest showed the highest accuracy from both figure 5.9 and figure 5.10. The kappa results also show that feature predications will be accurate in detecting both the positive and negative class.

5.3 Testing and Evaluation

For a machine learning model result to be accepted multiple evaluation techniques should be used. For the purpose of this project accuracy as well as confusion matrix, recall, precision and F-measure are used. From the results in section 5.2.3 random forest showed the best results for both accuracy and kappa, and is selected as the primary model for detecting encrypted traffic.

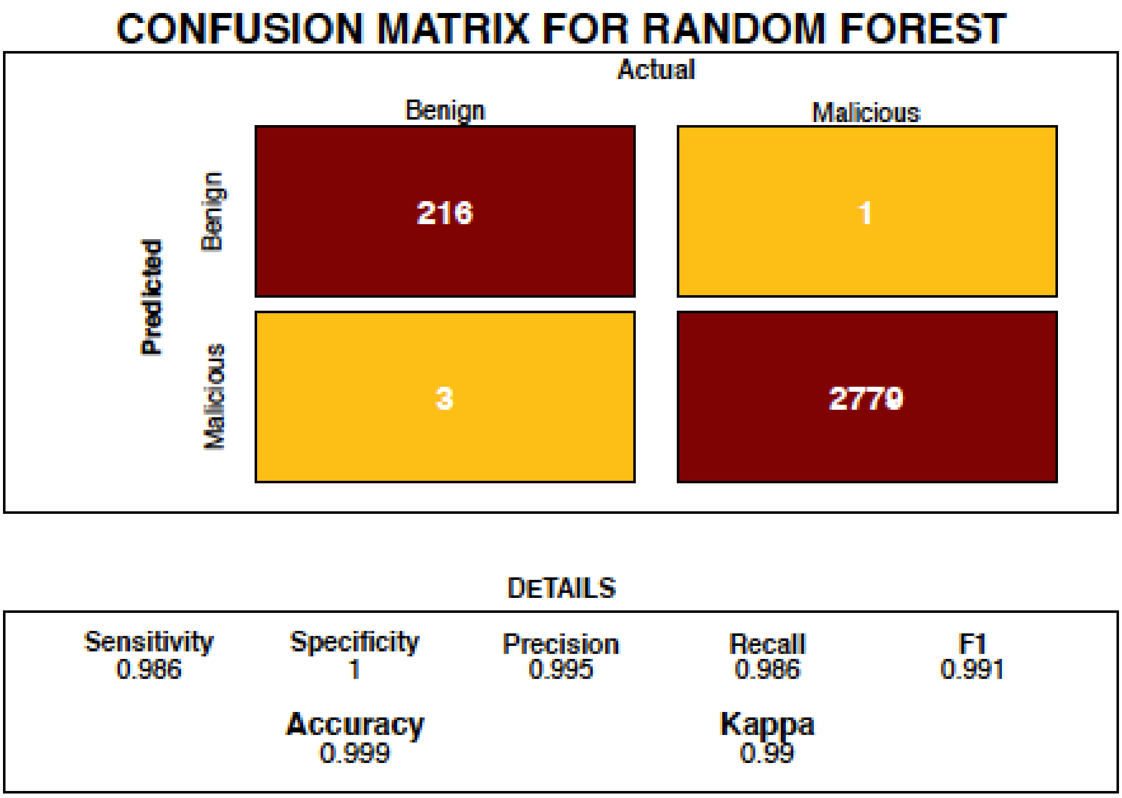

Figure 5.11: Evaluating Random Forest Encrypted Traffic Detection Model

The diagram shows a confusion matrix for the random forest model with maroon boxes shown True matches and yellow showing false matches. The other evaluation metrics support the accuracy of 99.9%. The results are also evaluated using Receiver Operating Characteristics (ROC) Area Under Curve (AUC) see Figure 5.12. The score for the AUC is 0.992970828816399.

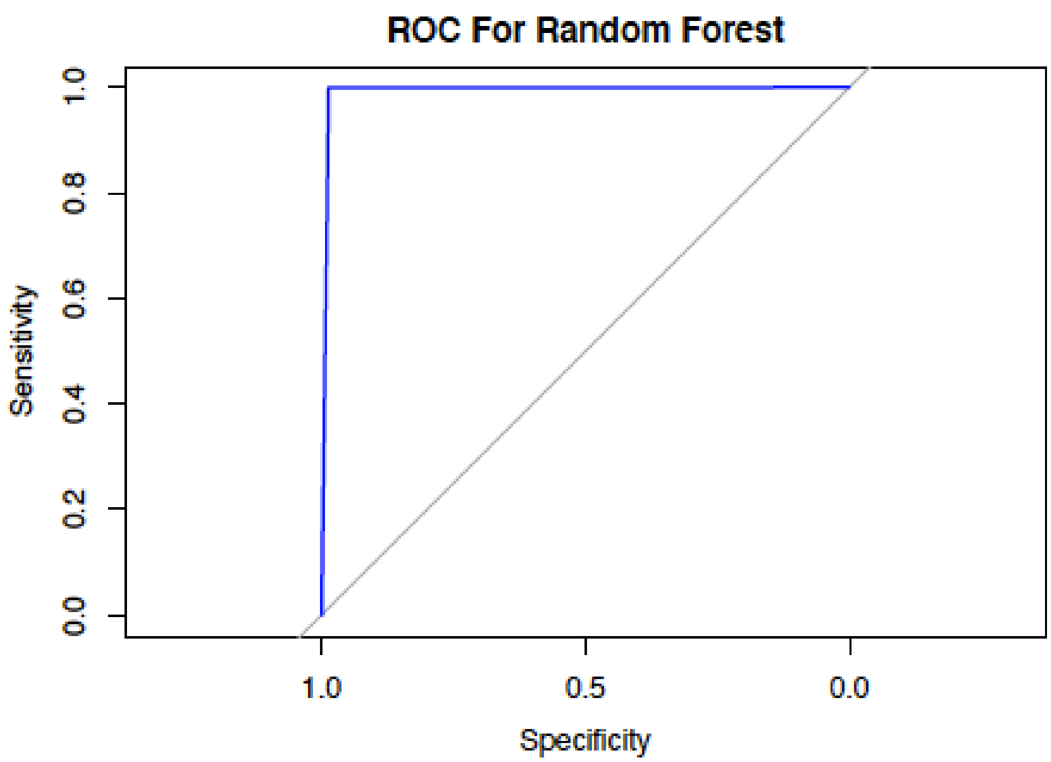

Figure 5.12: ROC Curve for Random Forest

5.4 Converting Solution to Application Programming Interface (API)

This section deals with presenting the stages involved in deploying the proposed solution to API. Figure 5.13 shows the steps involved.

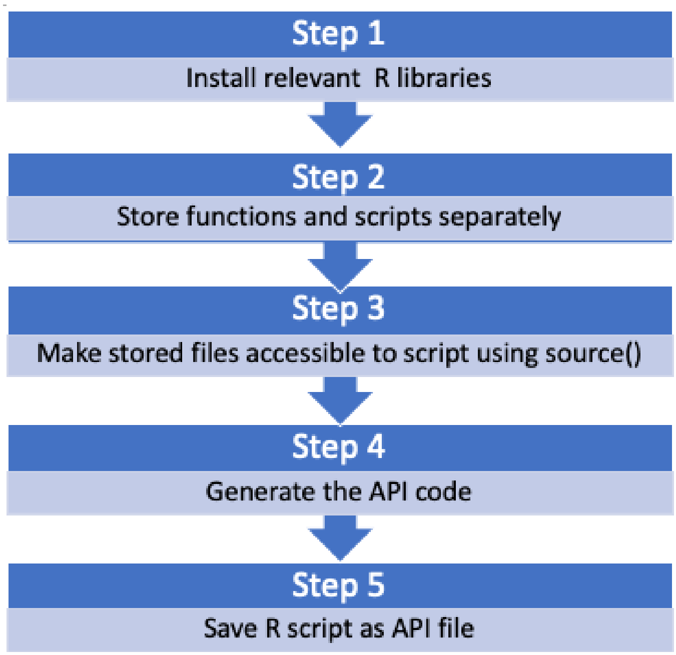

Figure 5.13: Steps in Converting Proposed Solution to API

CHAPTER 6: EVALUATION OF WORK

6.1 Introduction

This chapter presents the evaluation of the project's solution. This evaluation focuses on how the project has been able to meet up with the objectives of the study, project specification, comparison to the state-of-the-art and contribution to knowledge.

6.2 Evaluating Project Results with Existing Solutions

The output of this project is evaluated with related works. The related works highlighted in Table 6.1 utilized machine learning with encrypted traffic to build the model.

Table 6.1: Evaluation of proposed solution with Related Works

S/N	Author	Algorithm	Accuracy	F1 Score
1	Wang, Fok and Thing (2022)	Random Forest	94.71%	NA
2	Isingizwe et al. (2021)	Auto ML (7 models)	NA	NA
3	Proposed Solution	Random Forest	99.9%	0.991

6.3 Evaluating Project Results with Project Objective

This section presents an evaluation on how well this project has met with the objectives set out to achieve. The results are presented in Table 6.2.

Table 6.2: Evaluation of How Project Results Have Met Objectives

S/N	**Objective**	**How Project Met Objective**
1	To build a machine learning model for detecting HTTPS malware traffic without having to decrypt.	A Machine Learning model using random forest has been built and tested with completely encrypted traffic without the need of decryption during the detection process.
2	To validate the model and improve its performance through training and testing.	The Model is validated using multiple evaluation metrics including Kappa, F1 Score, recall, precision, confusion matrix and accuracy. The model was also

		improved through the evaluation of multiple algorithms for the best and evaluating the algorithms using statistical analysis.
3	To evaluate the model with other algorithms to assess the efficiency of the newly created system used in this research.	Four algorithms were selected after a comparative evaluation of related works. The final results were the best of the four algorithms and outperformed the state-of-the-art.
4	To document the system's testing, evaluation results and their applicability to the scientific repository of cybersecurity knowledge.	This report presents the stages in meeting up with the aim presenting a reliable, reproducible and scientifically acceptable approach to detecting encrypted malware traffic. The report also highlights the results, evaluation report and its applicability to cybersecurity as seen in the contribution to knowledge section

6.4 Evaluating Results with Respect to Functional Requirements

This section presents a tabular report on how well the project outcome has met with the functional requirements under the MoSCoW prioritization approach.

Table 6.3 Evaluation of Success rate for Functional Requirements

S/N	Requirements	MoSCoW	Comment
1	Four machine learning algorithms will be evaluated to select algorithm with the highest accuracy score.	Must	KNN, Decision Tree, SVM and Random Forest were used Random Forest was selected
2	Solution will analyze large amounts of data to identify patterns and make predictions	Should	A dataset of 2,097,149 was used in the first instance and 269,643 for the final instance. Features relevant to detecting malware were identified and presented in a

			correlation heatmap graph. Predictions were made using the test dataset.
3	Solution will have the ability to analyze encrypted traffic	Must	The dataset used contains fully encrypted traffic data and yielded 99.9% accuracy in detecting malicious traffic from benign traffic
4	Evaluation of the predicted classes must be confirmed using two evaluation metrics including accuracy and confusion matrix	Should	The predicted classes results was presented using accuracy score and confusion matrix
5	Integration with visualization and reporting tools.	Could	Visualization tools were used to present results and findings and also reported using r markdown presented as pdf.
6	Solution must present over 90% accuracy in detecting malware traffic	Must	The proposed model achieved an accuracy of 99.9% in detecting encrypted malware traffic
7	Recall, Precision and F-Measure should be used to confirm predicted classes	Should	Predicted classes were confirmed using Recall, Precision and f-Measure with scores 0.991, 1 and 0.995 respectively
8	Ability to handle missing or incomplete data	Could	The program is able to detect missing values and handle missing and incomplete data using tidyverse library
9	Integrate solution with physical devices and sensors	Won’t	NOT COVERED IN THIS PROJECT

10	Ability to perform automated data preprocessing and feature engineering	Must	The program provides data preprocessing functions and features engineering functions using tidyverse, corrplot and heatmaps.
11	Implementation of machine learning solution on Security information and event management (SIEM) systems	Won't	NOT COVERED IN THIS PROJECT
12	Put machine learning model into production through built API	Could	Not achieved due to time limitation
13	Deployment of solution as standalone desktop application.	Won't	NOT COVERED IN THIS PROJECT

Only one of the eleven function requirements divided into Must, Should, and Could, was not satisfied. The remaining tasks come under the Would not category, which was previously noted as being outside the study's purview.

6.5 Contributions to Knowledge

The following are contributions to knowledge recorded:

1. The project solution has shown very high accuracy in mitigating malware attacks even when the traffic is encrypted without affecting the privacy of the network users.
2. The model created when compared to the state-of-the-art shows better detection accuracy 99.9% as against 94.71% making a of 5.19% difference with lower cost as only one dataset is used and reduced chances of false positives as shown in the confusing matrix.
3. This project provides an effective machine learning approach to encrypted malware detection using PCAP files which are easily extractible from live networks using of-the-shelf and cloud-based tools like Wireshark and OpenStack.

From the four algorithms used Random Forest outperformed the others with very low variance in accuracy when the process is repeated making it the most stable algorithm for encrypted malware traffic detection. This was confirmed using a density plot and dot plot.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

The created model enhances the accuracy and effectiveness in detecting assaults even with encrypted packets based on the findings and outcome of the study on the detection of HTTPS malware traffic attacks without decrypting. The strategy lessens the difficulties associated with relying on false findings and signature-based solutions.

During the project, sourcing an encrypted dataset to train the model was challenging as most researches and available datasets dealt with unencrypted traffic. This challenge caused a major delay in meeting up with the aim of this project.

From the results recorded in this work, organizations that are vulnerable to malware traffic assaults should implement this solution as it offers a more effective alternative to current mitigation techniques employing encrypted communication. This strategy also saves money because it does not rely on the security infrastructure that is already in place.

Since research is an ongoing process that builds on prior work to address problems, this study and the methodology employed have demonstrated efficacy in the struggle against encrypted malware communications and served as a basis for further investigation.

7.2 Future Work

Although the assessment criteria employed indicated effective detection in the future, performance evaluation of this research was done through simulations when applying the suggested remedy. This study addresses an existing problem. The possibilities of these systems will be investigated in future work employing real-time data in computer networks and network devices.

Application Programming Interface (API) Representational State Transfer (REST) implementation of an enhanced detection model (API). Devices for intrusion detection and prevention, such as the System Information Event Management (SIEM) tool, may readily apply this.

This solution will be more effective for organisations and individuals regardless of location or services offered by using it with datasets from various areas, which will increase the solution's adoption rate.

Further model evaluation using created encrypted datasets generated under different conditions and environments with various types of attacks simulated is also an area for future research.

REFERENCE

ABOAOJA, F.A., ZAINAL, A., GHALEB, F.A., AL-RIMY, B.A.S., EISA, T.A.E. and ELNOUR, A.A.H., 2022. Malware detection issues, challenges, and future directions: A survey. *Applied Sciences,* **12**(17), pp. 8482.

AHMED, J., GHARAKHEILI, H.H., RUSSELL, C. and SIVARAMAN, V., 2022. Automatic Detection of DGA-Enabled Malware Using SDN and Traffic Behavioral Modeling. *IEEE Transactions on Network Science and Engineering,* **9**(4), pp. 2922-2939.

ALASHWALI, E.S., SZALACHOWSKI, P. and MARTIN, A., 2019. Does" www." Mean Better Transport Layer Security? *Proceedings of the 14th International Conference on Availability, Reliability and Security* 2019, pp. 1-7.

ALRAMMAL, M., NAVEED, M., SALLAM, S. and TSARAMIRSIS, G., 2022. A Two-Layered Machine Learning Approach for Anti-Malware Sustainability, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)* 2022, IEEE, pp. 7-11.

ANDERSON, B. and MCGREW, D., 2017. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity, *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining* 2017, pp. 1723-1732.

ANDERSON, B. and MCGREW, D., 2016. Identifying encrypted malware traffic with contextual flow data, *Proceedings of the 2016 ACM workshop on artificial intelligence and security* 2016, pp. 35-46.

ARGANDA-CARRERAS, I., KAYNIG, V., RUEDEN, C., ELICEIRI, K.W., SCHINDELIN, J., CARDONA, A. and SEBASTIAN SEUNG, H., 2017. Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics,* **33**(15), pp. 2424-2426.

ASONGU, S.A. and LE ROUX, S., 2017. Enhancing ICT for inclusive human development in Sub-Saharan Africa. *Technological Forecasting and Social Change,* **118**, pp. 44-54.

BADER, O., LICHY, A., HAJAJ, C., DUBIN, R. and DVIR, A., 2022. MalDIST: From encrypted traffic classification to malware traffic detection and classification, *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* 2022, IEEE, pp. 527-533.

BARUT, O., LUO, Y., LI, P. and ZHANG, T., 2022. R1dit: Privacy-preserving malware traffic classification with attention-based neural networks. *IEEE Transactions on Network and Service Management,* .

BOVENZI, G., CERASUOLO, F., MONTIERI, A., NASCITA, A., PERSICO, V. and PESCAPÉ, A., 2022. A comparison of machine and deep learning models for detection and classification of android malware traffic, *2022 IEEE Symposium on Computers and Communications (ISCC)* 2022, IEEE, pp. 1-6.

BOWLING, M., FÜRNKRANZ, J., GRAEPEL, T. and MUSICK, R., 2006. Machine learning and games. *Machine Learning,* **63**(3), pp. 211-215.

BYERS, R., WALTERMIRE, D. and TURNER, C., 2020. No title. *National Vulnerability Database (NVD) Metadata Submission Guidelines for Common Vulnerabilities and Exposures (CVE) Numbering Authorities (CNAs) and Authorized Data Publishers,* .

CELEBI, M.E. and AYDIN, K., 2016. *Unsupervised learning algorithms.* Springer.

CELIK, Z.B., WALLS, R.J., MCDANIEL, P. and SWAMI, A., 2015. Malware traffic detection using tamper resistant features, *MILCOM 2015-2015 IEEE Military Communications Conference* 2015, IEEE, pp. 330-335.

CHEN, M., HAO, Y., HWANG, K., WANG, L. and WANG, L., 2017. Disease prediction by machine learning over big data from healthcare communities. *Ieee Access,* **5**, pp. 8869-8879.

DENG, H., GUO, C., SHEN, G., CUI, Y. and PING, Y., 2023. MCTVD: A malware classification method based on three-channel visualization and deep learning. *Computers & Security,* **126**, pp. 103084.

DERI, L. and CARDIGLIANO, A., 2022. Using cyberscore for network traffic monitoring, *2022 IEEE International Conference on Cyber Security and Resilience (CSR)* 2022, IEEE, pp. 56-61.

DEVI, O.R., VALLABHANENI, S.P., HUSSAIN, M.A. and KUMAR, T.K., 2020. Security Analysis on Remote Authentication against Man-in-the-Middle Attack on Secure Socket Layer, *IOP Conference Series: Materials Science and Engineering* 2020, IOP Publishing, pp. 022015.

DORIGUZZI-CORIN, R. and SIRACUSA, D., 2022. FLAD: adaptive federated learning for DDoS attack detection. *arXiv preprint arXiv:2205.06661,* .

DUDDU, S., SOWJANYA, C.L., RAO, G.R. and SIDDABATTULA, K., 2020. Secure socket layer stripping attack using address resolution protocol spoofing, *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* 2020, IEEE, pp. 973-978.

DURUMERIC, Z., MA, Z., SPRINGALL, D., BARNES, R., SULLIVAN, N., BURSZTEIN, E., BAILEY, M., HALDERMAN, J.A. and PAXSON, V., 2017. The Security Impact of HTTPS Interception. *NDSS* 2017.

EHSAN, A., MAHMOOD, K., KHAN, Y.D., KHAN, S.A. and CHOU, K., 2018. A novel modeling in mathematical biology for classification of signal peptides. *Scientific reports,* **8**(1), pp. 1039.

EL BOUCHEFRY, K. and DE SOUZA, R.S., 2020. Learning in Big Data: Introduction to Machine Learning. *Knowledge Discovery in Big Data from Astronomy and Earth Observation.* Elsevier, pp. 225-249.

FINN, C., GOODFELLOW, I. and LEVINE, S., 2016. Unsupervised learning for physical interaction through video prediction, *Advances in neural information processing systems* 2016, pp. 64-72.

FRALEY, J.B., 2017. Improved Detection for Advanced Polymorphic Malware.

FU, Z., LIU, M., QIN, Y., ZHANG, J., ZOU, Y., YIN, Q., LI, Q. and DUAN, H., 2022. Encrypted Malware Traffic Detection via Graph-based Network Analysis, *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses* 2022, pp. 495-509.

GUERRERO-SAADE, J.A., 2022. HermeticWiper| New Destructive Malware Used in Cyber Attacks on Ukraine. *Sentinel Labs,* .

HAN, S., WU, Q., ZHANG, H. and QIN, B., 2022. Light-weight Unsupervised Anomaly Detection for Encrypted Malware Traffic, *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)* 2022, IEEE, pp. 206-213.

HARRINGTON, P., 2012. *Machine learning in action.* Manning Publications Co.

ISINGIZWE, D.F., WANG, M., LIU, W., WANG, D., WU, T. and LI, J., 2021. Analyzing learning-based encrypted malware traffic classification with automl, *2021 IEEE 21st International Conference on Communication Technology (ICCT)* 2021, IEEE, pp. 313-322.

KARA, I., 2022. Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges. *Expert Systems with Applications,* , pp. 119133.

KATAGIRI, N., 2022. Two explanations for the paucity of cyber-military, cross-domain operations. *Journal of Cybersecurity,* **8**(1), pp. tyac002.

KIM, D. and KIM, H.K., 2019. Automated dataset generation system for collaborative research of cyber threat analysis. *Security and Communication Networks,* **2019**, pp. 1-10.

KIM, G., CHOI, C. and CHOI, J., 2018. Ontology modeling for APT attack detection in an IoT-based power system, *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems* 2018, ACM, pp. 160-164.

KOHOUT, J., KOMÁREK, T., ČECH, P., BODNAR, J. and LOKOČ, J., 2018. Learning communication patterns for malware discovery in HTTPs data. *Expert Systems with Applications,* **101**, pp. 129-142.

KOUTSOKOSTAS, V., LYKOUSAS, N., APOSTOLOPOULOS, T., ORAZI, G., GHOSAL, A., CASINO, F., CONTI, M. and PATSAKIS, C., 2022. Invoice# 31415 attached: Automated analysis of malicious Microsoft Office documents. *Computers & Security,* **114**, pp. 102582.

KRAWCZYK, H., PATERSON, K.G. and WEE, H., 2013. On the security of the TLS protocol: A systematic analysis, *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I* 2013, Springer, pp. 429-448.

KROMBHOLZ, K., BUSSE, K., PFEFFER, K., SMITH, M. and VON ZEZSCHWITZ, E., 2019. " If HTTPS Were Secure, I Wouldn't Need 2FA"-End User and Administrator Mental Models of HTTPS, *2019 IEEE Symposium on Security and Privacy (SP)* 2019, IEEE, pp. 246-263.

KUMAR, A. and LIM, T.J., 2019. EDIMA: Early detection of IoT malware network activity using machine learning techniques, *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)* 2019, IEEE, pp. 289-294.

KUMAR, R. and SUBBIAH, G., 2022. Zero-day malware detection and effective malware analysis using Shapley ensemble boosting and bagging approach. *Sensors,* **22**(7), pp. 2798.

LETTERI, I., DELLA PENNA, G., DI VITA, L. and GRIFA, M.T., 2020. MTA-KDD'19: A Dataset for Malware Traffic Detection. *ITASEC* 2020, pp. 153-165.

LI, C., CHENG, Z., ZHU, H., WANG, L., LV, Q., WANG, Y., LI, N. and SUN, D., 2022. DMalNet: Dynamic malware analysis based on API feature engineering and graph learning. *Computers & Security,* **122**, pp. 102872.

LICHY, A., BADER, O., DUBIN, R., DVIR, A. and HAJAJ, C., 2023. When a RF beats a CNN and GRU, together—A comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification. *Computers & Security,* **124**, pp. 103000.

MAI, A., SCHEDLER, O., WEIPPL, E. and KROMBHOLZ, K., 2022. Are HTTPS Configurations Still a Challenge?: Validating Theories of Administrators' Difficulties with TLS Configurations, *HCI for Cybersecurity, Privacy and Trust: 4th International Conference, HCI-CPT 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26–July 1, 2022, Proceedings* 2022, Springer, pp. 173-193.

MANZIL, H.H.R. and NAIK, M.S., 2022. COVID-Themed Android Malware Analysis and Detection Framework Based on Permissions, *2022 International Conference for Advancement in Technology (ICONAT)* 2022, IEEE, pp. 1-5.

MARÍN, G., CAASAS, P. and CAPDEHOURAT, G., 2021. Deepmal-deep learning models for malware traffic detection and classification, *Data Science–Analytics and Applications: Proceedings of the 3rd International Data Science Conference–iDSC2020* 2021, Springer, pp. 105-112.

MAYER, R.E., 2019. Computer games in education. *Annual Review of Psychology,* **70**, pp. 531-549.

MITRE CORPORATION, 2021. Common Vulnerabilities and Exposures. *Mitre Corporation,* .

MOON, D., IM, H., LEE, J.D. and PARK, J.H., 2014. MLDS: multi-layer defense system for preventing advanced persistent threats. *Symmetry,* **6**(4), pp. 997-1010.

NICHO, M., OLUWASEGUN, A. and KAMOUN, F., 2018. Identifying vulnerabilities in apt attacks: A simulated approach, *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* 2018, IEEE, pp. 1-4.

PAIK, J., KIM, G., KANG, S., JIN, R. and CHO, E., 2022. Data Protection Based on Hidden Space in Windows Against Ransomware, *Proceedings of Sixth International Congress on Information and Communication Technology: ICICT 2021, London, Volume 1* 2022, Springer, pp. 629-637.

PRASSE, P., MACHLICA, L., PEVNÝ, T., HAVELKA, J. and SCHEFFER, T., 2017. Malware detection by analysing network traffic with neural networks, *2017 IEEE Security and Privacy Workshops (SPW)* 2017, IEEE, pp. 205-210.

RAJKOMAR, A., DEAN, J. and KOHANE, I., 2019. Machine learning in medicine. *New England Journal of Medicine,* **380**(14), pp. 1347-1358.

SATHISHKUMAR, B., SUNDARAVADIVAZHAGAN, B., MARTIN, B., SASI, G., CHANDRASEKAR, M., KUMAR, S.R., ELAMARAN, V., BALAJI, V. and ARUNKUMAR, N., 2020. Revisiting computer networking protocols by wireless sniffing on brain signal/image portals. *Neural Computing and Applications,* **32**, pp. 11097-11109.

SHARMA, P.K., MOON, S.Y., MOON, D. and PARK, J.H., 2017. DFA-AD: a distributed framework architecture for the detection of advanced persistent threats. *Cluster Computing,* **20**(1), pp. 597-609.

SHBAIR, W.M., CHOLEZ, T., GOICHOT, A. and CHRISMENT, I., 2015. Efficiently bypassing SNI-based HTTPS filtering, *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* 2015, IEEE, pp. 990-995.

SHIRE, R., SHIAELES, S., BENDIAB, K., GHITA, B. and KOLOKOTRONIS, N., 2019. Malware squid: A novel iot malware traffic analysis framework using convolutional neural network and binary visualisation, *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 19th International Conference, NEW2AN 2019, and 12th Conference, ruSMART 2019, St. Petersburg, Russia, August 26–28, 2019, Proceedings 19* 2019, Springer, pp. 65-76.

SILVA, J.C.F., TEIXEIRA, R.M., SILVA, F.F., BROMMONSCHENKEL, S.H. and FONTES, E.P., 2019. Machine learning approaches and their current application in plant molecular biology: A systematic review. *Plant Science,* **284**, pp. 37-47.

SİNGH, A., 2022. Classification of Malware in HTTPs Traffic Using Machine Learning Approach. *El-Cezeri,* **9**(2), pp. 644-655.

STEJSKAL, P. and FAIX, M., 2022. Legal Aspects of Misattribution Caused by Cyber Deception, *2022 14th International Conference on Cyber Conflict: Keep Moving!(CyCon)* 2022, IEEE, pp. 205-218.

TANG, Z., ZENG, X., GUO, Z. and SONG, M., 2020. Malware Traffic Classification Based on Recurrence Quantification Analysis. *Int.J.Netw.Secur.,* **22**(3), pp. 449-459.

TAYYAB, U., KHAN, F.B., DURAD, M.H., KHAN, A. and LEE, Y.S., 2022. A survey of the recent trends in deep learning based malware detection. *Journal of Cybersecurity and Privacy,* **2**(4), pp. 800-829.

VIRVILIS, N., GRITZALIS, D. and APOSTOLOPOULOS, T., 2013. Trusted Computing vs. Advanced Persistent Threats: Can a defender win this game? *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing* 2013, IEEE, pp. 396-403.

WANG, W., ZHU, M., ZENG, X., YE, X. and SHENG, Y., 2017. Malware traffic classification using convolutional neural network for representation learning, *2017 International conference on information networking (ICOIN)* 2017, IEEE, pp. 712-717.

WANG, Z., FOK, K.W. and THING, V.L., 2022. Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. *Computers & Security,* **113**, pp. 102542.

WANGEN, G., 2015. The role of malware in reported cyber espionage: a review of the impact and mechanism. *Information,* **6**(2), pp. 183-211.

WEITSCHEK, E., FISCON, G. and FELICI, G., 2014. Supervised DNA Barcodes species classification: analysis, comparisons and results. *BioData mining,* **7**(1), pp. 4.

ZHANG, M., WANG, L., JAJODIA, S. and SINGHAL, A., 2017. Evaluating the Network Diversity of Networks Against Zero-Day Attacks. *Network Security Metrics.* Springer, pp. 117-140.

APPENDIX 1

IMPLEMENTATION USING CICIDS2018 DATASET

```
data <- read.csv("CICIDS.csv", header = T)
library(tidyverse)
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0 v purrr 0.3.5
## v tibble 3.1.8 v dplyr 1.0.10
## v tidyr 1.2.1 v stringr 1.5.0
## v readr 2.1.3 v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()
data=filter(data, Label != "Label")
write.csv(data, "CICIDS_edited.csv", row.names=FALSE)
data <- read.csv("CICIDS_edited.csv", header = T)
```

Check if there are missing values

```
sum(is.na(data))
## [1] 1834
```

Checking the position of the of missing values

```
which(is.na(data))
## [1] 5298097 5298178 5298290 5298458 5298459 5298464 5298512 5298597 5298608
## [10] 5298677 5298729 5298878 5298894 5299004 5299014 5299120 5299273 5299289
## [19] 5299466 5299469 5299493 5299605 5299610 5299715 5299727 5299745 5299780
## [28] 5299894 5299911 5299974 5299982 5300027 5300035 5300039 5300065 5300235
## [37] 5300240 5300248 5300252 5300259 5300340 5300341 5300372 5300390 5300453
## [46] 5300523 5300560 5300562 5300585 5300588 5300602 5300695 5300696 5300765
## [55] 5300772 5301005 5301140 5301207 5301374 5301385 5301404 5301500 5301502
## [64] 5301565 5301595 5301606 5301626 5301650 5301677 5301753 5301798 5301811
## [73] 5301812 5301933 5302086 5302096 5302234 5302257 5302293 5302605 5302689
## [82] 5302704 5302717 5302718 5302722 5302801 5302853 5302877 5302906 5302924
## [91] 5302926 5302992 5302995 5303016 5303309 5303380 5303476 5303545 5303579
1
## [100] 5303604 5303615 5303710 5303717 5303728 5303733 5303798 5303846 5303853
## [109] 5304008 5304018 5304084 5304085 5304108 5304149 5304213 5304245 5304308
## [118] 5304349 5304425 5304429 5304448 5304458 5304510 5304511 5304521 5304526
## [127] 5304529 5304532 5304548 5304844 5304866 5305174 5305217 5305239 5305514
## [136] 5305533 5305536 5305615 5305752 5305776 5305779 5306826 5308076 5308696
## [145] 5308839 5309642 5309804 5310181 5310724 5311370 5311721 5312164 5313884
## [154] 5314013 5315069 5315132 5315372 5316631 5316876 5317551 5317562 5317563
## [163] 5317595 5317610 5317636 5317732 5317745 5317828 5317879 5317936 5317938
## [172] 5317952 5317990 5318003 5318005 5318023 5318024 5318123 5318124 5318133
## [181] 5318234 5318320 5318363 5318434 5318449 5318450 5318576 5318702 5318801
## [190] 5318835 5318982 5318984 5318999 5319081 5319099 5319166 5319170 5319232
## [199] 5319440 5319455 5319584 5319694 5319926 5319935 5319936 5320124 5320333
## [208] 5320483 5320485 5320486 5320674 5320716 5320772 5320869 5321109 5321110
## [217] 5321138 5321183 5321314 5321353 5321386 5321647 5321659 5321785 5322018
## [226] 5322027 5322033 5322257 5322274 5322276 5322277 5322325 5322326 5322410
## [235] 5322432 5322433 5322522 5322599 5322759 5322985 5323011 5323188 5323363
## [244] 5323386 5323517 5323574 5323582 5323591 5323630 5323741 5323747 5323805
## [253] 5323944 5324142 5324148 5324193 5324211 5324212 5324268 5324286 5324412
## [262] 5324536 5324755 5324756 5325310 5325596 5326633 5327374 5327386 5328230
## [271] 5328314 5330676 5331063 5331297 5331466 5332289 5333247 5333685 5334102
## [280] 5334249 5335315 5335600 5335744 5336062 5336197 5336646 5336695 5336696
## [289] 5336985 5336986 5336987 5337023 5337040 5337121 5337199 5337241 5337242
## [298] 5337254 5337275 5337276 5337323 5337365 5337376 5337445 5337599 5337702
## [307] 5337732 5337750 5337910 5337955 5338038 5338051 5338107 5338154 5338228
## [316] 5338235 5338334 5338343 5338634 5338647 5338657 5338795 5338811 5338823
```

```
## [325] 5338835 5338955 5338958 5339041 5339213 5339214 5339228 5339314 5339315
## [334] 5339330 5339540 5339653 5339680 5339713 5339753 5339782 5339789 5339790
## [343] 5339803 5339804 5339959 5339960 5340030 5340141 5340245 5340581 5340774
## [352] 5340782 5340791 5340971 5341011 5341205 5341206 5341219 5341388 5341503
## [361] 5341584 5341669 5341670 5341771 5341784 5341785 5341806 5341885 5342029
## [370] 5342159 5342162 5342170 5342182 5342199 5342392 5342417 5342421 5342479
## [379] 5342480 5342481 5342484 5342507 5342617 5342630 5342782 5342784 5342900
## [388] 5342925 5343012 5343226 5343239 5343242 5343257 5343284 5343511 5343560
## [397] 5343832 5344021 5344068 5344135 5344352 5344363 5344376 5344502 5344504
## [406] 5344571 5344603 5344830 5344868 5344871 5344970 5345011 5345013 5345736
## [415] 5345761 5346668 5346721 5346924 5346956 5347439 5347445 5348129 5348398
## [424] 5348568 5348723 5348755 5348801 5349531 5351457 5352370 5352668 5353420
## [433] 5353824 5353952 5354979 5355480 5356054 5356165 5356231 5358414 5358466
## [442] 5358618 5358702 5358822 5358945 5359017 5359034 5359087 5359137 5359319
## [451] 5359331 5359344 5359421 5359444 5359508 5359515 5359520 5359597 5359642
## [460] 5359662 5359844 5360274 5360322 5360419 5360420 5360663 5360704 5360738
## [469] 5360963 5360964 5360970 5361047 5361088 5361133 5361153 5361348 5361384
## [478] 5361453 5361502 5361600 5361637 5361668 5361673 5362041 5362089 5362309
## [487] 5362398 5362435 5362438 5362494 5362528 5362622 5362722 5362853 5362854
## [496] 5362870 5362871 5362872 5363406 5363429 5363504 5363514 5363516 5363687
## [505] 5363778 5363992 5364138 5364156 5364159 5364334 5364341 5364466 5364670
## [514] 5364687 5364688 5364716 5365074 5365076 5365212 5365392 5365657 5366041
## [523] 5366354 5366486 5366507 5366560 5366860 5367086 5367188 5367220 5368367
## [532] 5368392 5368799 5368930 5369038 5369179 5369608 5370185 5370377 5370478
## [541] 5370560 5370587 5370674 5371116 5371759 5371887 5372226 5373113 5373162
## [550] 5373723 5373979 5374049 5374077 5393281 5393798 5395184 5395185 5395206
## [559] 5395223 5395342 5395365 5395455 5395729 5396003 5396339 5396340 5396342
## [568] 5396374 5396946 5396948 5397395 5398357 5398378 5398380 5398381 5398382
## [577] 5398393 5398970 5399021 5399023 5399077 5399148 5399267 5399287 5399301
2
## [586] 5399303 5399480 5399485 5399742 5399747 5399777 5399783 5399787 5399947
## [595] 5399948 5399949 5399998 5400595 5400600 5400646 5400809 5400895 5401004
## [604] 5401032 5401041 5401042 5401070 5401202 5401207 5401642 5402075 5402190
## [613] 5402202 5402402 5402517 5402527 5402769 5403289 5403314 5403337 5403340
## [622] 5403381 5403396 5403399 5403630 5403631 5403632 5403633 5403634 5403754
## [631] 5403755 5403756 5403761 5403762 5403881 5404392 5404644 5404767 5404776
## [640] 5404993 5405294 5405311 5405995 5406163 5406222 5406224 5406237 5406431
## [649] 5406447 5406615 5406697 5406821 5406859 5406865 5406951 5407645 5407907
## [658] 5408271 5408476 5408497 5409370 5409371 5409392 5409423 5409430 5409438
## [667] 5409441 5409450 5409470 5409504 5409975 5410182 5410228 5410236 5410258
## [676] 5410268 5410361 5410456 5410603 5410623 5410662 5410712 5410716 5410727
## [685] 5410746 5411565 5411676 5411686 5411786 5412011 5412036 5412168 5412293
## [694] 5412608 5412669 5412692 5412710 5412774 5412803 5412805 5412807 5412821
## [703] 5412863 5413131 5413354 5413384 5413395 5413492 5413495 5413602 5413751
## [712] 5413761 5414115 5414251 5414335 5414686 5414927 5415169 5415173 5415182
## [721] 5415224 5415227 5415271 5415371 5415561 5415564 5415819 5415827 5415838
## [730] 5415846 5415887 5415909 5416340 5416469 5416472 5416476 5416716 5417176
## [739] 5417554 5417555 5417574 5417996 5418271 5418331 5418443 5418794 5418803
## [748] 5418804 5418831 5418841 5418842 5418854 5418856 5418931 5419663 5419882
## [757] 5419885 5420081 5420249 5420325 5420330 5420331 5420506 5420520 5420970
## [766] 5420977 5421092 5421094 5421232 5421293 5421375 5421376 5421635 5421646
## [775] 5421651 5421670 5421790 5421795 5421840 5421997 5421998 5422019 5422078
## [784] 5422106 5422134 5422990 5423689 5423756 5423992 5423998 5424132 5424163
## [793] 5424296 5424320 5424473 5424727 5424743 5424748 5424921 5424923 5424968
## [802] 5425463 5425507 5425557 5426018 5426048 5428583 5428584 5431122 5431197
## [811] 5431326 5431330 5431422 5431592 5431627 5431629 5432241 5432663 5432681
## [820] 5432896 5432974 5433279 5433309 5433334 5433355 5433514 5433896 5434354
## [829] 5434407 5434409 5434410 5434457 5434684 5434713 5434743 5435128 5435903
## [838] 5435918 5435943 5435958 5436006 5436012 5436033 5436056 5436108 5436110
## [847] 5436158 5438576 5438686 5438693 5438724 5438994 5439007 5439015 5439027
```

```
## [856] 5439165 5439178 5439180 5443326 5443598 5443768 5443847 5444097 5444098
## [865] 5444109 5444341 5444412 5445876 5445878 5446008 5446034 5446143 5446179
## [874] 5446386 5446387 5446391 5446392 5446555 5446558 5446569 5446570 5446571
## [883] 5446587 5446588 5446975 5448014 5448038 5448078 5448757 5449507 5449519
## [892] 5449707 5449713 5449734 5449735 5449967 5450628 5450641 5450643 5451654
## [901] 5451658 5451659 5451660 5451822 5452196 5453318 5453320 5453662 5453667
## [910] 5454129 5454174 5454202 5454294 5454399 5454509 5454528 5454534 5454535
## [919] 5454856 5454857 5454920 5454921 5455101 5455279 5455517 5456311 5456347
## [928] 5456375 5456661 5456666 5456683 5456706 5456747 5456886 5457075 5457076
## [937] 5457088 5457090 5457455 5457465 5457798 5458113 5458121 5458225 5458228
## [946] 5458399 5458408 5458663 5458672 5459212 5459215 5459224 5459284 5459357
## [955] 5460156 5460209 5460282 5460619 5460620 5460770 5461433 5461474 5461877
## [964] 5462001 5462148 5462152 5462206 5462599 5462616 5462634 5462655 5462768
## [973] 5462776 5465209 5465210 5465333 5465631 5465642 5465820 5465862 5465874
## [982] 5466176 5466350 5466500 5466802 5467008 5467307 5467516 5467528 5467738
## [991] 5467743 5467757 5467771 5467780 5467787 5467788 5467801 5467806 5467807
## [1000] 5467809 5468010 5468209 5468253 5469258 5469579 5469627 5471735 5471736
## [1009] 5472118 5472308 5472513 5472519 5472534 5472535 5473136 5473309 5473486
## [1018] 5473510 5473520 5473587 5473598 5473718 5474827 5475071 5475234 5475244
## [1027] 5475431 5475459 5475463 5475465 5475473 5475509 5475518 5475553 5475564
## [1036] 5475742 5475743 5475794 5475796 5475797 5476449 5476607 5477334 5477340
## [1045] 5477346 5478149 5478163 5478171 5478332 5478515 5478555 5478566 5478577
## [1054] 5478914 5478925 5479397 5479411 5479511 5479526 5479532 5480054 5480230
## [1063] 5480856 5480973 5481152 5481171 5481221 5481222 5481270 5481272 5481675
3
## [1072] 5481699 5481711 5482237 5482322 5482929 5483783 5483891 5483919 5483954
## [1081] 5484075 5484297 5484307 5484542 5484753 5484884 5484896 5484900 5484910
## [1090] 5484929 5484950 5484959 5485155 5485908 5485982 5486059 5486229 5486240
## [1099] 5486241 5486273 5486324 5486390 5486457 5486460 5486830 5486837 5486894
## [1108] 5487143 5487232 5487319 5487835 5488098 5488192 5488263 5488594 5488640
## [1117] 5488712 5489041 5489267 5489317 5489325 5489326 5489485 5490876 5490881
## [1126] 5490885 5491217 5492152 5492162 5492164 5492375 5492441 5492453 5492555
## [1135] 5493003 5493170 5493171 5493792 5494029 5494030 5494187 5494492 5494493
## [1144] 5494506 5494525 5494535 5494616 5494956 5495458 5495579 5495763 5496592
## [1153] 5496596 5496894 5497228 5497289 5497426 5497464 5497501 5497519 5497524
## [1162] 5497872 5497949 5498080 5498174 5498218 5498842 5498868 5499337 5499387
## [1171] 5499577 5499580 5500052 5500270 5500318 5500335 5500556 5500557 5500843
## [1180] 5500857 5500912 5500913 5500914 5500916 5500938 5500960 5500990 5501388
## [1189] 5501702 5501717 5501748 5502047 5502182 5502511 5502956 5502957 5503385
## [1198] 5503388 5503712 5504198 5504204 5504211 5504927 5505145 5505411 5505773
## [1207] 5506079 5506094 5506227 5506228 5506229 5506230 5506232 5506333 5506347
## [1216] 5506387 5506388 5506414 5506563 5506689 5507235 5507642 5507664 5508277
## [1225] 5508284 5508340 5508464 5508528 5508533 5508551 5508585 5508681 5509015
## [1234] 5509846 5509904 5510679 5511288 5511644 5511654 5511699 5512069 5512153
## [1243] 5512202 5512250 5512251 5512261 5512276 5512280 5512297 5512298 5512330
## [1252] 5512335 5512336 5512339 5513762 5513783 5513790 5514083 5514116 5516142
## [1261] 5516225 5516836 5516844 5516845 5516933 5517292 5517628 5517640 5517814
## [1270] 5518059 5518071 5518087 5518366 5518372 5518398 5518414 5518415 5518501
## [1279] 5518502 5518504 5518863 5519162 5519228 5519245 5519408 5519483 5519492
## [1288] 5519501 5519502 5519847 5519967 5520052 5520054 5520264 5520677 5520934
## [1297] 5521516 5521530 5521536 5522347 5522396 5524116 5524995 5524997 5524998
## [1306] 5525029 5525070 5525633 5525649 5525660 5525842 5526145 5526290 5526740
## [1315] 5526757 5526758 5526764 5526771 5526873 5526883 5526891 5526892 5526893
## [1324] 5527011 5527130 5527133 5527139 5527143 5527159 5527727 5527860 5527861
## [1333] 5527862 5527863 5528366 5528404 5528655 5528715 5528739 5528748 5528760
## [1342] 5529132 5529353 5529585 5529657 5529849 5529872 5530152 5530168 5530213
## [1351] 5530232 5530431 5530673 5530685 5530690 5530954 5530968 5530974 5531169
## [1360] 5531820 5531843 5531853 5531986 5531996 5532265 5532266 5532284 5532287
## [1369] 5532294 5532334 5532426 5532609 5532840 5532871 5534222 5534495 5534957
## [1378] 5535046 5535070 5535087 5535089 5535109 5535219 5535307 5535308 5535371
```

```
## [1387] 5535725 5535763 5535839 5535915 5535925 5536108 5536226 5536227 5536229
## [1396] 5537038 5537049 5537054 5537151 5537469 5537480 5537707 5537873 5537875
## [1405] 5537951 5537993 5538260 5538294 5538323 5538354 5538562 5538830 5538841
## [1414] 5538949 5538998 5539537 5539552 5539559 5539573 5539588 5539676 5540141
## [1423] 5540193 5540194 5540244 5540359 5540869 5540982 5540997 5541197 5541309
## [1432] 5541328 5541351 5541430 5541593 5541596 5541597 5541984 5542049 5542067
## [1441] 5542797 5542804 5543064 5543565 5543566 5543576 5543577 5543794 5543848
## [1450] 5543929 5543931 5543943 5543947 5543969 5543970 5544267 5544350 5544730
## [1459] 5545446 5545472 5546413 5546862 5546896 5546927 5547589 5547629 5547630
## [1468] 5547633 5547646 5547659 5547662 5547692 5547713 5548065 5548077 5548194
## [1477] 5548496 5548933 5548949 5549299 5549631 5549708 5549795 5549833 5549852
## [1486] 5549853 5549855 5549867 5550056 5550078 5550107 5550113 5550704 5551209
## [1495] 5551547 5551577 5551642 5551878 5552154 5552302 5552313 5552314 5552332
## [1504] 5552555 5552620 5552629 5552663 5552668 5552890 5552900 5552902 5552949
## [1513] 5553135 5553170 5553958 5553993 5554006 5554412 5554542 5554688 5554908
## [1522] 5554917 5554967 5555063 5555108 5555259 5555261 5555379 5555389 5555390
## [1531] 5555886 5555933 5556318 5556347 5556348 5556416 5556418 5556730 5557464
## [1540] 5557525 5558320 5558346 5559203 5559215 5559231 5559233 5562035 5562041
## [1549] 5562423 5562534 5562592 5562595 5562605 5562654 5562829 5563046 5563067
4
## [1558] 5563074 5563549 5563655 5564746 5564747 5564763 5564811 5565169 5565414
## [1567] 5565473 5565887 5565899 5565905 5568230 5568329 5568399 5568410 5568455
## [1576] 5568456 5568457 5568465 5568530 5568538 5568549 5568675 5570251 5570278
## [1585] 5570758 5573012 5573016 5573342 5573686 5573878 5573879 5573949 5574035
## [1594] 5574052 5574449 5574462 5574463 5574599 5574631 5574632 5574633 5574768
## [1603] 5574769 5574948 5575074 5575125 5575178 5575310 5575912 5576135 5576314
## [1612] 5576315 5576528 5576538 5576547 5576552 5576561 5576606 5576836 5576837
## [1621] 5576844 5576845 5576867 5576968 5577391 5579734 5579743 5579972 5580860
## [1630] 5581895 5582179 5582263 5582684 5583015 5583103 5583132 5583147 5583148
## [1639] 5583149 5585736 5585737 5585745 5585898 5587398 5587447 5587682 5590307
## [1648] 5590360 5590511 5590650 5590708 5590721 5590730 5590731 5591559 5591608
## [1657] 5591663 5592225 5592395 5592503 5592589 5592634 5592744 5592795 5593148
## [1666] 5593602 5593926 5594002 5594049 5594168 5594170 5594388 5594561 5594598
## [1675] 5594599 5594610 5594948 5595301 5595302 5595392 5595516 5595544 5596353
## [1684] 5596466 5596501 5596513 5596760 5596766 5597043 5597045 5597149 5597241
## [1693] 5597242 5597286 5597406 5598241 5598261 5599279 5599440 5599465 5599608
## [1702] 5599747 5599750 5599751 5601077 5601079 5601150 5602217 5602333 5602370
## [1711] 5602662 5602855 5603054 5603236 5603418 5603436 5603454 5603464 5603794
## [1720] 5604236 5604238 5604379 5604769 5604779 5604830 5605037 5605077 5605473
## [1729] 5605482 5605483 5605595 5605596 5605751 5605787 5605892 5605893 5605905
## [1738] 5605907 5605913 5605923 5606379 5606468 5606544 5606759 5606962 5606977
## [1747] 5606987 5607273 5607781 5608787 5608811 5609119 5609194 5609515 5609831
## [1756] 5609873 5609888 5610721 5611470 5612043 5612046 5612055 5612056 5612762
## [1765] 5613008 5613452 5613453 5613672 5613747 5613980 5614117 5614417 5614525
## [1774] 5614563 5614792 5615024 5615025 5615120 5615483 5615621 5615929 5615930
## [1783] 5616074 5616097 5616423 5616441 5616442 5616483 5616498 5616501 5616725
## [1792] 5617043 5617164 5617166 5617358 5617410 5617503 5617588 5617651 5618121
## [1801] 5618164 5618675 5618680 5618695 5619229 5619484 5619716 5620913 5621071
## [1810] 5621084 5621094 5621105 5621106 5621111 5621112 5621619 5621997 5622434
## [1819] 5623480 5623653 5623655 5623672 5623743 5624138 5624336 5625179 5625200
## [1828] 5625211 5626069 5626084 5626287 5626685 5627474 5628410
str(data)
## 'data.frame': 331100 obs. of 80 variables:
## $ Dst.Port : int 0 0 67 0 0 0 67 0 0 0 ...
## $ Protocol : int 0 0 17 0 0 0 17 0 0 0 ...
## $ Timestamp : chr "01/03/2018 08:17:11" "01/03/2018 08:20:07" "01/03/2018 08:17:18" "01/03/2018 ##
$ Flow.Duration : int 115307855 60997457 61149019 60997555 61997503 61997503 92344822 60997542
60997552 ## $ Tot.Fwd.Pkts : int 5 2 5 2 3 3 9 2 2 2 ...
## $ Tot.Bwd.Pkts : int 0 0 0 0 0 0 0 0 0 0 ...
## $ TotLen.Fwd.Pkts : int 0 0 1500 0 0 0 2700 0 0 0 ...
```

$ TotLen.Bwd.Pkts : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Pkt.Len.Max : int 0 0 300 0 0 0 300 0 0 0 ...
$ Fwd.Pkt.Len.Min : int 0 0 300 0 0 0 300 0 0 0 ...
$ Fwd.Pkt.Len.Mean : num 0 0 300 0 0 0 300 0 0 0 ...
$ Fwd.Pkt.Len.Std : num 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Pkt.Len.Max : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Pkt.Len.Min : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Pkt.Len.Mean : num 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Pkt.Len.Std : num 0 0 0 0 0 0 0 0 0 0 ...
$ Flow.Byts.s : num 0 0 24.5 0 0 ...
$ Flow.Pkts.s : num 0.0434 0.0328 0.0818 0.0328 0.0484 ...
$ Flow.IAT.Mean : num 28800000 61000000 15300000 61000000 31000000 31000000 11500000 61000000 61000000 5
$ Flow.IAT.Std : num 32400000 0 12800000 0 42400000 ...
$ Flow.IAT.Max : int 61000000 61000000 32600000 61000000 61000000 61000000 31300000 61000000 61000000 ## $ Flow.IAT.Min : int 812396 61000000 3530939 61000000 999909 1000011 3999448 61000000 61000000 ## $ Fwd.IAT.Tot : int 115000000 61000000 61100000 61000000 62000000 62000000 92300000 61000000 61000000 ## $ Fwd.IAT.Mean : num 28800000 61000000 15300000 61000000 31000000 31000000 11500000 61000000 61000000 ## $ Fwd.IAT.Std : num 32400000 0 12800000 0 42400000 ...
$ Fwd.IAT.Max : int 61000000 61000000 32600000 61000000 61000000 61000000 31300000 61000000 61000000 ## $ Fwd.IAT.Min : int 812396 61000000 3530939 61000000 999909 1000011 3999448 61000000 61000000 ## $ Bwd.IAT.Tot : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.IAT.Mean : num 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.IAT.Std : num 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.IAT.Max : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.IAT.Min : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.PSH.Flags : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.PSH.Flags : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.URG.Flags : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.URG.Flags : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Header.Len : int 0 0 40 0 0 0 72 0 0 0 ...
$ Bwd.Header.Len : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Pkts.s : num 0.0434 0.0328 0.0818 0.0328 0.0484 ...
$ Bwd.Pkts.s : num 0 0 0 0 0 0 0 0 0 0 ...
$ Pkt.Len.Min : int 0 0 300 0 0 0 300 0 0 0 ...
$ Pkt.Len.Max : int 0 0 300 0 0 0 300 0 0 0 ...
$ Pkt.Len.Mean : num 0 0 300 0 0 0 300 0 0 0 ...
$ Pkt.Len.Std : num 0 0 0 0 0 0 0 0 0 0 ...
$ Pkt.Len.Var : num 0 0 0 0 0 0 0 0 0 0 ...
$ FIN.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ SYN.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ RST.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ PSH.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ ACK.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ URG.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ CWE.Flag.Count : int 0 0 0 0 0 0 0 0 0 0 ...
$ ECE.Flag.Cnt : int 0 0 0 0 0 0 0 0 0 0 ...
$ Down.Up.Ratio : int 0 0 0 0 0 0 0 0 0 0 ...
$ Pkt.Size.Avg : num 0 0 360 0 0 ...
$ Fwd.Seg.Size.Avg : num 0 0 300 0 0 0 300 0 0 0 ...
$ Bwd.Seg.Size.Avg : num 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Byts.b.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Pkts.b.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Fwd.Blk.Rate.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Byts.b.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Pkts.b.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Bwd.Blk.Rate.Avg : int 0 0 0 0 0 0 0 0 0 0 ...
$ Subflow.Fwd.Pkts : int 5 2 5 2 3 3 9 2 2 2 ...
$ Subflow.Fwd.Byts : int 0 0 1500 0 0 0 2700 0 0 0 ...
$ Subflow.Bwd.Pkts : int 0 0 0 0 0 0 0 0 0 0 ...

```
## $ Subflow.Bwd.Byts : int 0 0 0 0 0 0 0 0 0 0 ...
## $ Init.Fwd.Win.Byts: int -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
## $ Init.Bwd.Win.Byts: int -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
## $ Fwd.Act.Data.Pkts: int 0 0 4 0 0 0 8 0 0 0 ...
## $ Fwd.Seg.Size.Min : int 0 0 8 0 0 0 8 0 0 0 ...
## $ Active.Mean : num 1812348 0 3530939 0 999909 ...
## $ Active.Std : num 0 0 0 0 0 ...
```
6
```
## $ Active.Max : int 1812348 0 3530939 0 999909 0 8749231 0 0 0 ...
## $ Active.Min : int 1812348 0 3530939 0 999909 0 4608317 0 0 0 ...
## $ Idle.Mean : num 56700000 61000000 19200000 61000000 61000000 61000000 15800000 61000000
61000000 ## $ Idle.Std : num 6010058 0 12500000 0 0 ...
## $ Idle.Max : int 61000000 61000000 32600000 61000000 61000000 61000000 31300000 61000000
61000000 ## $ Idle.Min : int 52500000 61000000 7999725 61000000 61000000 61000000 7468536 61000000
61000000 ## $ Label : chr "Benign" "Benign" "Benign" "Benign" ...
data=select(data, -Timestamp)
```
Handling missing values and conversion of Label to factor
```
data= data[complete.cases(data), ]
data[sapply(data, simplify = 'matrix', is.infinite)] <- 0
data <- na.omit(data)
data[, colSums(data != 0) > 0]
data[1:78] <- lapply(data[1:78], as.numeric)
names(data[, sapply(data[1:78], function(v) var(v, na.rm=TRUE)==0)])
## [1] "Bwd.PSH.Flags" "Bwd.URG.Flags" "Fwd.Byts.b.Avg" "Fwd.Pkts.b.Avg"
## [5] "Fwd.Blk.Rate.Avg" "Bwd.Byts.b.Avg" "Bwd.Pkts.b.Avg" "Bwd.Blk.Rate.Avg"
data=data[,sapply(data[1:78], function(v) var(v, na.rm=TRUE)!=0)]
data <- data[sample(nrow(data), 20000), ]
dim(data)
## [1] 20000 71
Dataset_all.pca <- prcomp(data[,1:70], center = TRUE,scale. = TRUE)
Data_std_dev <- Dataset_all.pca$sdev
Data_pr_var <- Data_std_dev^2
Data_prop_varex <- Data_pr_var/sum(Data_pr_var)
plot(Data_prop_varex, xlab = "Principal Component", ylab = "Proportion of Variance",type = "b")
```
7

0 10 20 30 40 50 60 70

0.00 0.05 0.10 0.15 0.20

Principal Component

Proportion of Variance
```
Modified_features<-Dataset_all.pca$x[,1:40]
Modified_features<-as.data.frame(Modified_features)
Modified_features$lbl<-as.factor(data$Label)
```
Split dataset to training and testing dataset using 80:20 ratio
```
library(caret)
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
## lift
index <- createDataPartition(Modified_features$lbl, p=0.80, list=FALSE)
Test_dataset <- Modified_features[-index,]
Train_dataset <- Modified_features[index,]
dim(Test_dataset)
## [1] 4000 41
```
8
```
dim(Train_dataset)
## [1] 16000 41
```
Training model using decision tree algorithm

```
library(ranger)
library(caret)
fit.DT <- train(lbl~., data=Train_dataset, method="rpart")
fit.KNN <- train(lbl~., data=Train_dataset, method="knn")
fit.SVM <-train(lbl~., data=Train_dataset, method = "svmLinear")
fit.RF   <- train(as.factor(lbl)~., data=Train_dataset, method = "ranger")
Combined_results <- resamples(list(DT=fit.DT, KNN=fit.KNN,SVM=fit.SVM, RF=fit.RF))
summary(Combined_results)
## Call:
## summary.resamples(object = Combined_results)
##
## Models: DT, KNN, SVM, RF
## Number of resamples: 25
##
## Accuracy
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## DT 0.7527556 0.7602359 0.7644474 0.7638169 0.7662667 0.7731749 0
## KNN 0.7286888 0.7346592 0.7388263 0.7390028 0.7427795 0.7538197 0
## SVM 0.7490579 0.7571477 0.7605900 0.7603831 0.7636177 0.7729988 0
## RF    0.7016482 0.7150538 0.7190509 0.7198805 0.7262664 0.7326531 0
## Kappa
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## DT 0.1937313 0.2059985 0.2147247 0.2154369 0.2219679 0.2612129 0
## KNN 0.2371047 0.2568370 0.2637522 0.2643029 0.2708873 0.2924334 0
## SVM 0.2214274 0.2379003 0.2430096 0.2444804 0.2507479 0.2715362 0
## RF 0.2041923 0.2341986 0.2413481 0.2430946 0.2532174 0.2756860 0
```

Prediction Results

```
Model_prediction <- predict(fit.DT, Test_dataset)
confusionMatrix(Model_prediction, Test_dataset$lbl)
## Confusion Matrix and Statistics
##
## Reference
## Prediction Benign Infilteration
## Benign 2863 920
## Infilteration 27 190
##
## Accuracy : 0.7632
9
## 95% CI : (0.7498, 0.7764)
## No Information Rate : 0.7225
## P-Value [Acc > NIR] : 2.782e-09
##
## Kappa : 0.2151
##
## Mcnemar's Test P-Value : < 2.2e-16
##
## Sensitivity : 0.9907
## Specificity : 0.1712
## Pos Pred Value : 0.7568
## Neg Pred Value : 0.8756
## Prevalence : 0.7225
## Detection Rate : 0.7157
## Detection Prevalence : 0.9457
## Balanced Accuracy : 0.5809
##
## 'Positive' Class : Benign
##
```