Republic of Iraq

Ministry of Higher Education and Scientific

Research University of Babylon

College of Information Technology

Software Department

Dynamic Machine Learning Algorithm for Multiple Malware Detection

A Thesis
Submitted to the Council of the College of Information Technology for Postgraduate Studies of the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology – Software

By

Zahraa Najah Fadhil Lafta

Supervised by

Prof. Dr. Wesam Sameer Bhaya

2023A.D. **1445A.H.**

بسم الله الرحمن الرحيم

﴿ٱلرَّحۡمَٰنُ ۞ عَلَّمَ ٱلۡقُرۡءَانَ ۞ خَلَقَ
ٱلۡإِنسَٰنَ ۞ عَلَّمَهُ ٱلۡبَيَانَ ۞ ٱلشَّمۡسُ وَٱلۡقَمَرُ
بِحُسۡبَانٍ﴾

صدق الله العظيم

سورة الرحمن

Supervisor Certification

I certify that the thesis entitled “**Dynamic Machine Learning Algorithm for Multiple Malware Detection**” was prepared under my supervision at the department of Software / College of Information Technology / the University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology - Software.

Signature:

Supervisor Name: Prof. Dr. Wesam Sameer Bhaya

Date: 1 /10 / 2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled “**Dynamic Machine Learning Algorithm for Multiple Malware Detection**” for debate by the examination committee.

Signature: Prof. Dr. Ahmed Saleem Abbas

Head of Software Department

Date: 1/10 / 2023

Certification of the Examination Committee

We, the undersigned, certify that (**Zahraa Najah Fadhil Lafta**) candidate for the degree of Master in Information Technology - Software, has presented his thesis of the following title (**Dynamic Machine Learning Algorithm for Multiple Malware Detection**) as it appears on the title page and front cover of the thesis that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:

Signature:	Signature:
Name: Dr. Hawraa Hassan Abbas	Name: Dr. Hiba Mohammed Jaafar
Title: : Professor	Title: Lecturer
Date: 1/10 / 2023	Date: 1/10 /2023
(**Chairman**)	(**Member**)

Signature:	Signature:
Name: Ali Kadhim Bermani	Name: Dr. Wesam Sameer Bhaya
Title: Assistant Professor	Title: Professor
Date: 1/10 / 2023	Date: 1/10 /2023
(**Member**)	(**Member**) // Main supervisor

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Wesam Sameer Bhaya
Title: Professor
Date: 1/10 / 2023
(**Dean of Collage of Information Technology**)

Declaration

I hereby declare that this thesis, submitted to University of Babylon in partial fulfillment of requirement for the degree of Master in Information
Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source are appropriately cited in the references.

Signature:
Name: Zahraa Najah Fadhil
Date: 1 / 10 /2023

Dedication

To the one who spent his life for us to be here

..My father..

To the light that illuminates the darkness of my life

..My mother..

To those who stood to help and support me in anytime and anywhere

.. My husband, brothers, and sisters..

Acknowledgments

Foremost, I am highly grateful to **Allah** for His unlimited blessings that continue to flow into my life, and because of You, I made this through against all odds.

To my supervisor, Dr. **Wesam Sameer Bhaya**: many thanks for all your advice and comments during our research career. I am grateful for your help to make this research in the best possible form and make me an academic who deserves this certificate.

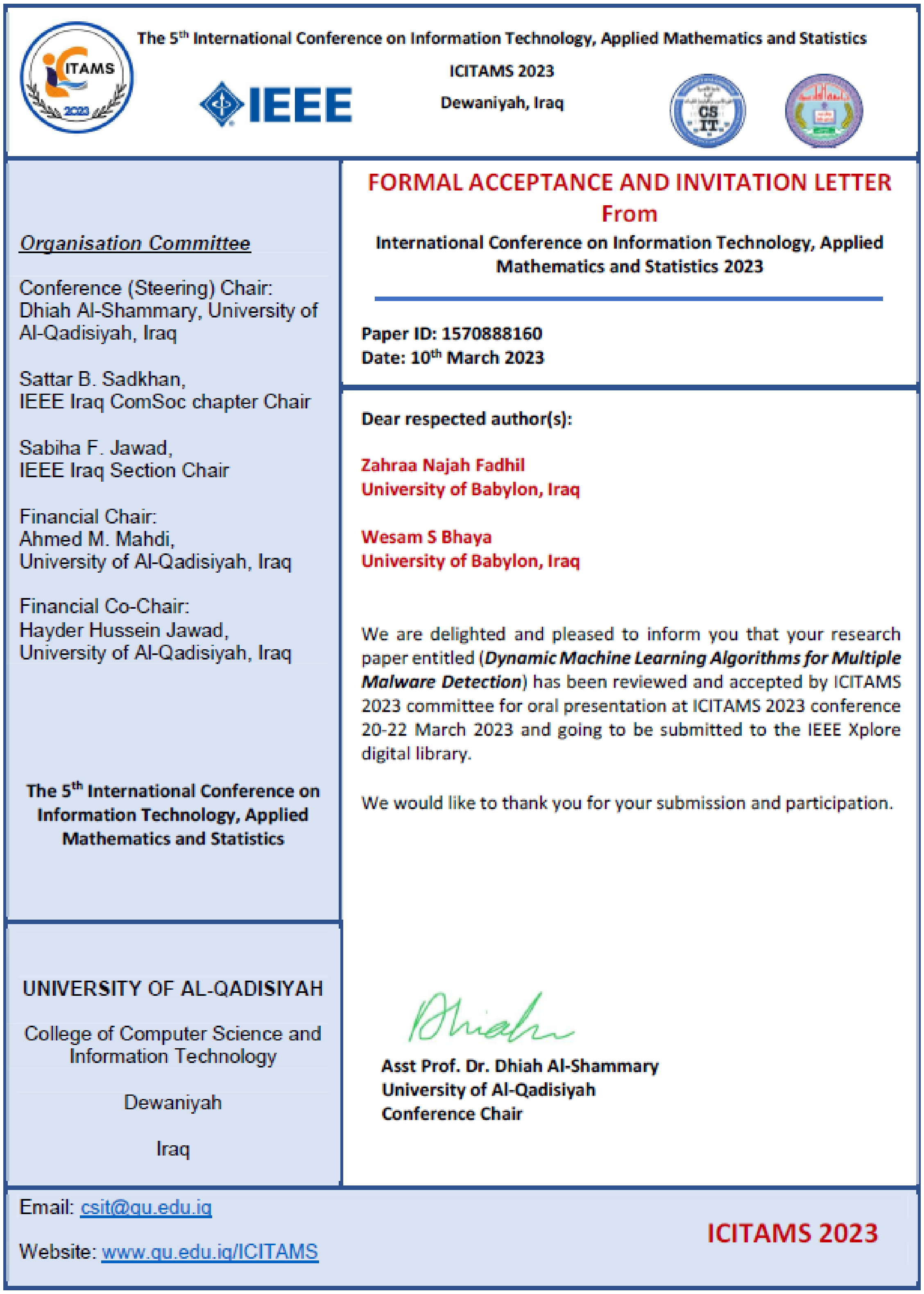
To the inhalation and exhalation which I cannot live without them .. **my parents**... Thank you for every minute of your life that you sacrificed for me.

For my support in this life, my husband **Nashwan**, who did not leave me in the most difficult circumstances. Thanks for everything.

Unlimited thanks to my **brothers and sisters** who did not stop supporting and helping me throughout this period.

Last but not least, I would like to thank all the kind, helpful and lovely people who helped me directly or indirectly to complete this work and apologize to them for not being able to mention them by name here, but they are in my heart.

Zahraa Najah Fadhil



The 5th International Conference on Information Technology, Applied Mathematics and Statistics

ICITAMS 2023

Dewaniyah, Iraq

FORMAL ACCEPTANCE AND INVITATION LETTER

From

International Conference on Information Technology, Applied Mathematics and Statistics 2023

Paper ID: 1570888160
Date: 10th March 2023

Dear respected author(s):

Zahraa Najah Fadhil
University of Babylon, Iraq

Wesam S Bhaya
University of Babylon, Iraq

We are delighted and pleased to inform you that your research paper entitled (***Dynamic Machine Learning Algorithms for Multiple Malware Detection***) has been reviewed and accepted by ICITAMS 2023 committee for oral presentation at ICITAMS 2023 conference 20-22 March 2023 and going to be submitted to the IEEE Xplore digital library.

We would like to thank you for your submission and participation.

Dhiah

Asst Prof. Dr. Dhiah Al-Shammary
University of Al-Qadisiyah
Conference Chair

Organisation Committee

Conference (Steering) Chair:
Dhiah Al-Shammary, University of Al-Qadisiyah, Iraq

Sattar B. Sadkhan,
IEEE Iraq ComSoc chapter Chair

Sabiha F. Jawad,
IEEE Iraq Section Chair

Financial Chair:
Ahmed M. Mahdi,
University of Al-Qadisiyah, Iraq

Financial Co-Chair:
Hayder Hussein Jawad,
University of Al-Qadisiyah, Iraq

The 5th International Conference on Information Technology, Applied Mathematics and Statistics

UNIVERSITY OF AL-QADISIYAH

College of Computer Science and Information Technology

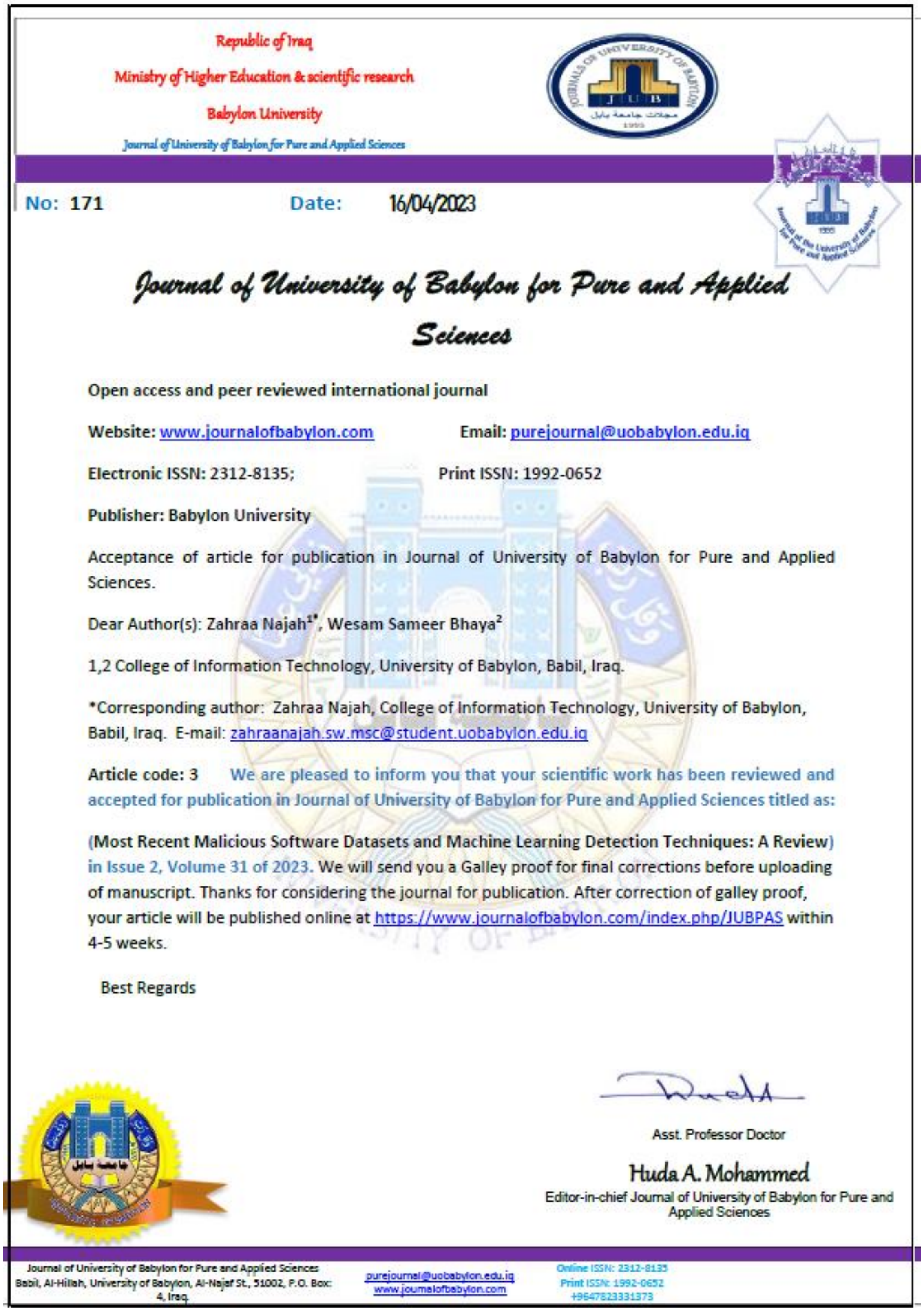
Dewaniyah

Iraq

Email: csit@qu.edu.iq

Website: www.qu.edu.iq/ICITAMS

ICITAMS 2023



Republic of Iraq

Ministry of Higher Education & scientific research

Babylon University

Journal of University of Babylon for Pure and Applied Sciences

No: 171 Date: 16/04/2023

Journal of University of Babylon for Pure and Applied Sciences

Open access and peer reviewed international journal

Website: www.journalofbabylon.com Email: purejournal@uobabylon.edu.iq

Electronic ISSN: 2312-8135; Print ISSN: 1992-0652

Publisher: Babylon University

Acceptance of article for publication in Journal of University of Babylon for Pure and Applied Sciences.

Dear Author(s): Zahraa Najah[1*], Wesam Sameer Bhaya[2]

1,2 College of Information Technology, University of Babylon, Babil, Iraq.

*Corresponding author: Zahraa Najah, College of Information Technology, University of Babylon, Babil, Iraq. E-mail: zahraanajah.sw.msc@student.uobabylon.edu.iq

Article code: 3 We are pleased to inform you that your scientific work has been reviewed and accepted for publication in Journal of University of Babylon for Pure and Applied Sciences titled as:

(Most Recent Malicious Software Datasets and Machine Learning Detection Techniques: A Review) in Issue 2, Volume 31 of 2023. We will send you a Galley proof for final corrections before uploading of manuscript. Thanks for considering the journal for publication. After correction of galley proof, your article will be published online at https://www.journalofbabylon.com/index.php/JUBPAS within 4-5 weeks.

Best Regards

[illegible]

Asst. Professor Doctor

Huda A. Mohammed

Editor-in-chief Journal of University of Babylon for Pure and Applied Sciences

Journal of University of Babylon for Pure and Applied Sciences Babil, Al-Hillah, University of Babylon, Al-Najaf St., 51002, P.O. Box: 4, Iraq.

purejournal@uobabylon.edu.iq www.journalofbabylon.com

Online ISSN: 2312-8135 Print ISSN: 1992-0652 +9647823331373

Abstract

As the number of malware attacks continues to increase, there is a growing need for effective and efficient malware detection techniques. Dynamic machine learning algorithms have emerged as a promising approach for detecting malware in real-time. These algorithms leverage the dynamic behavior of malware to distinguish it from benign software.

This thesis uses publicly available DDoS 2019 and malware analysis datasets and the Microsoft Malware Classification Database (BIG 2015) and uses more than one method to select the feature in the dataset such as additional tree feature selection and correlation feature selection. After evaluating the effectiveness of Decision Trees (DT), Feedforward Neural Networks (FFNN), and Support Vector Machines (SVM)for detecting multiple types of malware based on their performance on these datasets, a hard voting method was utilized to choose the best classification algorithm that provides the highest predictive rate for all types of malware detection, in order to create a dynamic selection classification system, and this proposal achieved a success rate of 99.99%.

The Decision Tree approach achieved a 100% success rate, the FFNN approach achieved a 99.99% success rate, and the SVM technology achieved a 99.52% success rate, according to the DDoS 2019 dataset. The larger 2015 dataset showed that the DT method had a success rate of 98.17%, the FFNN approach had a success rate of 96.89%, and the SVM technology had an accuracy of 93.20%. For the malware analysis dataset, the DT approach achieved a success rate of 90.77%, the FFNN approach had a success rate of 88.88%, and the SVM method achieved a success rate of only 80.0%.

Table of Contents

Table of Tables

Table of Figures

Table of Algorithms

List of Abbreviations

Abbreviation	Description
AMZD	Advanced on Min-Max Z-score Decimal scaling
CIC	Canadian Institute for Cybersecurity
CNNs	Convolution Neural Networks
CV	Computer Viruses
DDoS	Distributed Denial of Service
DT	Dissection Tree
ET	Extra Trees Classifier feature selection
E-Voting	Electronic Voting
FFNN	Feed Forward neural network
FN	False Negative
FP	False Positive
HPCs	High Performance Counters
IOCs	indicators of compromise
KDD	Knowledge Discovery and Data Mining(data set)
KNNs	k-Nearest Neighbors
LDAs	Linear Discriminant Analyses
LPPMs	Location-Privacy Preserving Mechanisms
LSTMs	Long Short-Term Memories
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNB	Multinomial Naive Bayes
NB	Naive base
NN	Neural Network
RBF	Radial Basis Function
RF	Random force
SDN	Software-Defined Networking
SGB	Sovereign Gold Bond Scheme
SMO	Spider Monkey Optimization
SVM	Support vector machine
TN	True Negative
TP	True Positive

Chapter one
General Introduction

1.1 Introduction

In general, security is defined as "the characteristic or situation of being secure—being devoid of risk"[1]. In other words, the goal is to guard against adversaries—those who would cause harm, intentionally or unintentionally. Malicious software is introduced as any sort of programmer that is designed to damage or hack the user. They may be seeking to steal your information, or they may be acting maliciously. In any case, it's hardly worth the effort to speculate on a hacker's intentions[1]. Malware is described as unauthorized software meant to infiltrate or harm a computer system. Malware is the umbrella term for all types of computer dangers. A straightforward taxonomy of malware includes file infectors and standalone malware. Malware can also be classified depending on their specific behavior: worms, backdoors, Trojans, rootkits, spyware, adware, etc. It has become commonplace for cybercriminals to conceal their malicious operations within the devices of their victims, making it harder for antivirus and other security software to detect them. Scientists see the need for improved malware detection and prevention systems, thus they develop innovative techniques, methodologies, and tools [2].

There are three fundamental ways for detecting malware: signature-based, anomaly-based, and heuristic. Almost all antivirus software uses signature-based detection and prevention techniques nowadays. Antivirus software examines programmers for unusual activity by comparing them to known malware patterns[3]. Anomaly-based approaches, on the other hand, focus on observing usual system activity to detect deviations. Moreover, the most recent technique utilizes Heuristics. When it comes to recognizing and diagnosing dangerous and abnormal behaviors in the system, researchers adopt Heuristic-based tactics to maximize the earlier types by employing new advanced algorithms and processes such as machine learning and neural networks [4] .

However, malware detection through standard, signature-based methods is becoming increasingly difficult as all current malware applications tend to have multiple polymorphic layers to avoid detection or to use side mechanisms to automatically update to a newer version at short intervals to avoid detection by any antivirus software. Therefore, machine learning techniques have so been implemented by researchers in an effort to identify malware[5].

Over the past two decades, machine learning has been a pillar of information technology, bringing with it a rather major, if occasionally hidden, component of our lives. There is reason to anticipate that smart data analysis will become more commonplace as a critical component for technological advancement given the ever-increasing volumes of data that are available[6]. The study of how to design computers to learn from and adapt to new data, such as that acquired by sensors or stored in databases, is known as machine learning, a subfield of artificial intelligence. Information may aid a learner in identifying key characteristics of a probability distribution that would otherwise be obscure. Information can be thought of as examples that demonstrate relationships between observables[7]. One of the main objectives of machine learning research is to automatically learn to recognize complex patterns and make decisions based on data. detection of cyberattacks through deep learning and machine learning methods. Support vector machines (SVMs), k-nearest neighbors (KNNs), linear discriminant analysis (LDAs), long short-term memory (LSTMs), convolution neural networks (CNNs), and auto encoders are among the methods used for malware identification most frequently [8] [9].

1.2 Problem Statement

As the number of destructive attacks and their targets increase, they become more difficult to identify and prevent as a result of their changing behavior. Most of the traditional malware detection systems are based on statistical, analytical, and machine learning models. Virus signature methods are often used in malware detection to protect against malware, but one of their drawbacks is the temporary matching of the traditional virus signatures they are in. The eigenvalue is too long, and it cannot meet the needs of information security[10].

The bulk of antivirus programmers depends on regular expressions and patterns to categorize malware. Since file properties must be changed to detect and prevent newly created malware, antivirus programmers are less likely to update their databases to identify and prevent malware. The production of attack signatures required almost maximum human effort. Signature-based antivirus solutions recognize malicious and benign Windows PE32 executables. Binary categorization uses signatures, partial matching, regex, or heuristics. This clustering approach is no longer appropriate since malware diversification randomly diversifies code and data to reduce mutant similarity[11].

1.3 Aims and Objectives

The aim of the thesis is to develop a dynamic selection classification system that can effectively detect multiple types of malicious threats by adapting and choosing the best machine learning algorithms.

The objectives of this thesis are to achieve the aim of detecting higher accuracy of different data threats by determining the most effective machine learning classification model for detecting malware.

The following are the study's precise objectives:

1. Use more than one method to select the feature in the dataset such as Extra tree feature selection and Correlation feature selection.
2. Developing a dynamic selection classification system that utilizes a hard voting approach to select the best algorithm for detecting multiple types of malware based on their specific characteristics and behaviors.

1.4 Related Works

In this section, some previous studies on multiple machine learning algorithms and malware detection are listed.

In 2021, Bawazeer et. al. [2], proposed a Machine Learning (ML) algorithms increasingly use High Performance Counters (HPCs) events for malware detection . A variety of high-performance counters (HPCs) are available in contemporary processors for measuring and tracking process events like as memory accesses, instructions, etc. during their execution. This research presents an analytical analysis to categorize the HPC-based machine learning techniques that have been utilized to detect malware. The most effective and practical aspects of HPCs that have been used to spot unusual activity on various systems are also emphasized. Additionally, a number of research from the scientific literature are mimicked using the neural network (NN) algorithms Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Full Order Radial Basis Function (RBF). According to the simulation findings, the accuracy of MLP, CNN, and Full Order RBF are 96.95 percent, 98.22 percent, and 98.68 percent, respectively.

In 2019, Atallah [7], proposed a majority voting ensemble method that is able to predict the possible presence of heart disease in humans. The forecast is based on straightforward, reasonably priced medical exams performed at any nearby clinic. Since the model was trained using actual data from healthy and unwell patients, the project's goal is to increase the trust and accuracy of the doctor's diagnosis. In order to deliver more accurate results than using only one model, the model categorizes the patient based on the consensus of numerous machine learning models. Finally, using the hard voting ensemble model, our method generated an accuracy of 90%.

In 2021, Uchenna et.al. [8], proposed a provide exhaustive analytical approaches and cutting-edge solutions for addressing the malware threat in IoT applications. Particularly, static, dynamic, and hybrid analyses have been used by researchers to address the security vulnerabilities facing several IoT systems. The utility of the given analytical methodologies was demonstrated using case studies involving smart home systems, smart factories, smart devices, and IoT application protocols. Systems like 6thSense, a complete context-aware architecture for sensing security in IoT devices, have an accuracy and F-score of about 97%. A similar defense against white-box attacks is offered by location-privacy preserving mechanisms (LPPMs) with integrated targeted maneuvers, which reduce the likelihood of success to 3%.

In 2021, Baek et. al. [9], proposed a 2-MaD method is devised to protect smart city IoT devices against sophisticated attacks. In order to discover misclassified malware in the second stage, 2-MaD performed dynamic analysis on files that static analysis had deemed harmless. Compared to static analysis-based malware detection (94.46%), 2-MaD offers a greater detection accuracy. The FNR of 2-MAD was 0.019 as opposed to 0.062 for dynamic analysis. Due to the required size of dynamic analysis features for computing speed, the 2-MaD model has caused a loss of features in this work. A feature selection method that employs hashing weights and the TF-IDF will be further investigated.

In 2022, Akhtar et. al. [10], proposed an increasing interest in ML-based malware detection. They developed a defense that evaluated machine learning algorithms for malware detection and selected the best one. The detection accuracy of DT (99%), CNN (98.76%), and SVM (96.41%) is high. The effectiveness of the DT, CNN, and SVM algorithms in detecting malware was tested on a particular dataset (DT = 2.01%, CNN = 3.97%, and

SVM = 4.63%). A machine learning (ML) classifier that used static analysis to extract attributes from PE data was compared to two other ML classifiers in this experiment. Machine learning systems can recognize hazardous data thanks to our research. The highest level of accuracy we've seen is 99.9% with the DT machine learning system. Testing has shown that static analysis based on PE data and carefully selected data produces positive results.

In 2022, Alshamrani [11], proposed an ML model capable of identifying JavaScript and malicious API call assaults in PDF files is presented. This study also examined the performance of other different classifiers, including DT, RF, LR, SVC, and SGB, using the dataset. The best outcomes came from the RF classifiers used in this investigation. This approach is 4 percent more effective than the most recent PDF classifiers when compared to other PDF classifiers, with an F1-score of 0.986. Functionality to run an object scanner inside the PDF document is integrated to further reinforce the system's defense against malicious code obfuscation techniques. This functionality looks for any unprocessed objects. By employing this technique, harmful code included in unparsed objects can be easily identified and removed. Future plans call for expanding compatibility with more file types. To gain a deeper comprehension of texts, employ a creative data mining method.

In 2019, Ismail et.al. [12], proposed an outline of the application of machine learning technology for the detection of obfuscated computer viruses and suggested an alternative to the conventional signature-based detection method. There are 20 instances of correctly identified normal files (TN) and 20 instances of correctly identified computer viruses (CV), according to the confusion matrix results. This shows that even if the test set includes malware files that have been encoded, the SMO classifier model can accurately classify every case. The following test sets and the categorization results follow the same procedure. The table illustrates that the majority of test sets including computer viruses may be correctly classified. Based on the 5 test sets, an average of only 1 out of 25 obfuscated virus files was mistakenly identified as normal, resulting in a 99.5% accuracy.

In 2020, Rios et.al. [13], proposed a three methods: Fuzzy Logic (FL), MLP, and Euclidean Distance (ED) for detecting Distributed Denial of Service (DDoS) attacks. Distributed Denial of Service (DDoS) attacks are still among the most dangerous attacks on the Internet. To investigate the detection of this type of attack, we evaluate and compare the use of four machine learning algorithms: Multi-Layer Perceptron (MLP) neural network with backpropagation, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB), among the four Machine Learning algorithms, the best classification results are obtained with MLP, which, for emulated traffic, leads to an F1-score of 98.04% for attack traffic .

In 2020, Santos et.al. [14], proposed an analysis of the problem and implementation of four machine learning algorithms (SVM, MLP, Decision Tree, and Random Forest) with the purpose of classifying DDoS attacks in an SDN-simulated environment as a result, the best accuracy with the

Random Forest algorithm and the best processing time with the Decision Tree algorithm. The DT, RF, MLP, and SVM algorithms were evaluated in identifying attacks (DT =0.99 %, RF = 1.00%, MLP =0.98 %, and SVM =0.92 %).

Table 1.1 A summary of the related works

No	year	Purpose	Datasets	Techniques	Findings
[2]	2021	The aim is to provide an analytical study of the classification of machine learning algorithms based on the HPCs that have been used to detect malware.	Specific dataset.	1- (MLP) 2- (CNN) 3- (RBF)	The accuracy of MLP, CNN, and Full Order RBF are 96.95 percent, 98.22 percent, and 98.68 percent, respectively.
[7]	2019	proposed a majority voting ensemble method that is able to predict the possible presence of heart disease in humans.	real-life data	voting ensemble method	The accuracy of 90% based on the hard voting ensemble model.
[8]	2021	The main goal is to provide comprehensive analytics methods and cutting-edge solutions to address the malware threat in IoT applications.	Specific dataset.	location-privacy preserving mechanisms (LPPMs)	The accuracy and F-score of systems are roughly 97%.
[9]	2021	The aim is to be designed to protect IoT devices in	The data for the	2-MaD method	2-MaD has a higher detection accuracy

		smart cities from sophisticated attacks and perform dynamic analysis on files classified as harmless through static analysis, enabling misclassified malware to be found.	analysis of the video content was gathered by extracting videos using various		than static analysis-based malware detection (94.46%).
[10]	2022	proposed an increasing interest in ML-based malware detection.	Specific dataset.	1- SVM 2-DT 3-CNN	DT (99%), CNN (98.76%), and SVM (96.41%) have high detection accuracy.
[11]	2022	proposed an ML model capable of identifying JavaScript and malicious API call assaults in PDF files is presented.	Specific dataset.	DT, RF, LR, SVC, and SGB	Comparing this method to other PDF classifiers shows that it has a high F1-score of 0.986.
[12]	2019	proposed an outlined the application of machine learning technology for the detection of obfuscated computer viruses	Specific dataset.	SMO classifier model	The accuracy of 99.5%.
[13]	2020	proposed a three methods: Fuzzy Logic (FL), MLP, and Euclidean Distance (ED) for detecting Distributed Denial of Service (DDoS) attacks.	real traffic datasets	(MLP) (K-NN), (SVM), (MNB),	The best classification results are obtained with MLP, which, for emulated traffic, leads to an F1-score of 98.04% for attack traffic .

[14]	2020	proposed an analysis of the problem and implementation of four machine learning algorithms with the purpose of classifying DDoS attacks in an SDN-simulated environment.	DDoS dataset.	SVM, RF, DT,MLP	1-DT =0.99 % 2- RF=1.00% 3-MLP =0.98 % 4-SVM =0.92 %

1.5 Thesis layout

Additionally, to this chapter, four chapters will be presented, these are:

Chapter Two reviews an extensive description of the main concepts of Malware Detection as well as Dynamic Machine Learning Algorithms which are used in this thesis.

Chapter Three presents the proposed system, and practical stages for the suggested system

Chapter Four clarifies the implementation of the proposed methodology in the (DDoS 2019 dataset, Big 2015 dataset, and Malware Analysis dataset) and the experimental results of this implementation are obtained.

Chapter Five draws conclusions that have been reached through this thesis and gives suggestions for future work.

Chapter Two
Theoretical Background

2.1 Information Security

The era of information has arrived. Every aspect of our lives requires keeping records, Or to put it another way, knowledge is an asset just like any other. Information must be safeguarded against interference for its own sake. Secrecy (Confidentiality), immutability (Integrity), and availability (when necessary) to the proper party are the three pillars of information security [16].Only authorized and dependable company employees were given access to the file in order to maintain the confidentiality of the information it contained. Similarly, only persons who have been given permission may edit the information included in the files. By designating a single point of contact, we made sure the files would always be available. The introduction of computers did not change the three essential safeguards. It is crucial to safeguard the confidentiality, integrity, and usability of data stored on computers. The application of these criteria, however, is particular and more challenging[17]. The use of networks and communications technologies to transfer data between computers as well as between computers and between terminal users is the second significant change that has had an impact on security. For the security of data transmission, network security measures must be implemented. The term "network security" may be deceptive because almost all organizations, governments, and educational institutions connect their computer systems to a web of other networks. "Internet security" is a term that is widely used to describe the procedures carried out to protect such a computer network[18].

The distinctions between network security and computer security are hazy. The computer virus is a well-known type of cyberattack. A physical method of infection occurs when a diskette or optical disc with a virus is obtained and then used to infect a computer. It's also feasible for viruses to spread over the internet. Once the virus has infected a computer, it must be removed using either internal computer security tools or external anti-virus software. Modern computers are extremely powerful, operating with pinpoint accuracy and unfathomably rapid speeds. The main goals of information security are to avoid losing the confidentiality, integrity, and availability of systems and data. The majority of security measures and controls are designed to avoid losses in one or more of these categories[19].

2.2 Malware

Malicious code, also referred to as malware, is destructive code that is put into legitimate applications for evil motives. Concerns concerning consumer privacy have been raised as a result of the growth of Internet-connected heterogeneous devices, which has altered the danger environment. the most typical method by which harmful software enters computers undetected by their owners. Free downloads of software like games, web browsers, antivirus programs, etc. are frequently used to spread malicious software. Due to its widespread use for financial transactions, the Internet has resulted in significant losses for both organizations and individuals[20].

Many hackers have been drawn to creating malware because it is so profitable. Polymorphic and metamorphic malware can be generally categorized into two groups, and both are challenging, if not impossible, to identify with a signature-based detector. Authors of malware frequently employ techniques like instruction permutation, register reassignment, code permutation with conditional instructions, no-operation insertion, etc. to create new iterations. Malware analysis is used to look at and understand destructive behaviour. Extraction of strings, opcodes, bytes of code, API/system calls, and a network trace are common steps in malware analysis[21].

In particular, malware refers to programmers who are intended to damage the confidentiality, integrity, or availability of a computer system. Malware is a term for numerous types of destructive software. Since the adverse influence may utilize executable code or interpreted code, scripts, macros, etc., the term "software" should be used in the broadest sense conceivable in this context[22]. Malware frequently bears the name of the computer system whose security settings were allegedly hacked. The list of targets may or may not be explicitly known depending on the infection. Malware is frequently divided into a number of types based on the way it enters the target system and the type of policy violation it is intended to cause. The following terms will be used[23] :

- Computer viruses are harmful programs that reproduce and propagate to other computers by contaminating files that are sent from infected to uninfected computers. A human user who is not vigilant may transmit a CD-ROM with the infected file, or the virus may start the transport by sending the infected file as an e-mail attachment[24].
- A worm is a type of malicious software that spreads from infected system to infected machine over a network without infecting any files in the process.

- A Trojan horse is a piece of software that entices the user to run it while hiding its destructive or hazardous intent. The user could experience a wide range of issues as a result of the payload, including the deletion of their data or the installation of further harmful or unwanted software. Trojan horses called droppers are used to "inject" the worm into users' local networks and start a worm outbreak. Trojan horses, which are camouflaged as seemingly helpful programs that consumers download from the Internet, are frequently used to distribute spyware. The spyware is installed simultaneously with the program and is packed with it. Those spyware developers who make an effort to do the right thing by the law may include a licensing agreement with vague language describing the spyware's activity, which users are very unlikely to read and comprehend.
- Malware known as "logic bombs" is activated when a specific need is satisfied, such as when a specific date or time has passed or when a specific piece of information, such as a file or database item, has been added or removed.
- Rabbit: (also known as bacterium) Malware that completely exhausts a system's supply of a certain kind of resource, such as message buffers, file space, or process control blocks.

- Malicious software that allows an attacker to access a target system without going through the standard login and authentication procedures is referred to as a "backdoor"[25].

2.3 Malicious Software and Machine Learning

Due to its global distribution, diversity, and complexity, malicious software poses a very tough threat to networks and computers. Because malware is constantly evolving, cyber security professionals must constantly improve their defenses. Malware spread quickly due to the polymorphic and metamorphic strategies it used to avoid detection and hide its true intentions. A polymorphic engine modifies the code of polymorphic malware while keeping the pathogen's original functionality[26]. Encryption and packing are the two most common ways to hide a code. Packers' compression methods cover up the original source code. The original code is then recreated in memory and used by the unpacking methods during runtime. In order to thwart attempts at analysis, crypters are programs that encrypt and otherwise modify malware or specific portions of its code[27].

An executable's signature, which functions similarly to a fingerprint, is one of its unique features. However, signature-based systems are unable to detect novel malware types. Security experts proposed behavior-based detection as a solution to these issues; although time-consuming, this method analyses a file's features and behavior to ascertain whether it is malicious. In order to get beyond the limitations of conventional antivirus engines and keep up with new threats and variants, researchers have started utilizing machine learning as a supplement to their solutions. Over the past ten years, there has been an upsurge in research into and application of machine learning techniques for malware identification and categorization. The success and consolidation of machine learning approaches would not have been feasible prior to the convergence of three recent advances. Over the past few decades, the field of machine learning has grown more quickly, achieving ground-breaking accuracy and scalability outcomes on a range of applications, including computer vision, speech recognition, and natural language processing. A machine learning workflow is a cycle that starts with data collection and progresses through data preparation and cleaning, model creation, model validation, and production deployment, as shown in Figure 2.1[28].

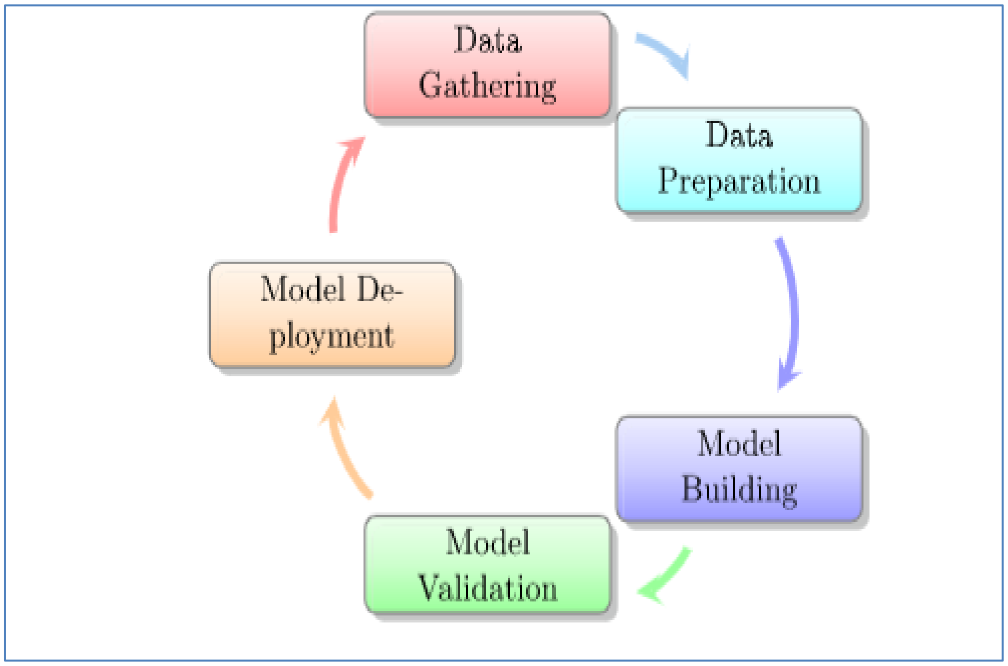


Figure 2.1 Machine learning workflow[28].

2.4 Data Preprocessing

Data preprocessing methods are important to prepare the dataset. In general, all of these methods fall into two categories: selecting data objects and attributes for the analysis or creating/changing the attributes. These methods include several strategies for handling dataset issues such as noise, missing values, and inconsistent data.

In all cases, the goals of preprocessing are[29] :

- ✓ Reducing the dataset size to achieve a more efficient analysis concerning time, cost, and quality.
- ✓ Adapting the dataset to best suit the selected analysis method.

In this thesis, many steps were followed to prepare the research data for the prediction model[29].

2.4.1 Data Cleaning

There are a variety of factors at play here. Because electronic health record data were acquired for reasons other than those of the research, there is a possibility that certain clinical characteristics were not consistently recorded, leading to missing data. Similarly, imperfect or limited tools used to collect the data (such as the attenuation of blood pressure measurements measured through an arterial line) or human mistake in data input may lead to noisy data. All of the above may potentially cause discrepancies in the numbers. All of this adds up to the need of taking careful data cleansing measures before any analysis is performed . There are two methods to do data cleaning[29].

- **Missing values**

Data analysis involves missing values or data. These missing numbers are frequently ascribed to human mistake while processing data, equipment failure, respondents refusing to answer particular questions, study drop-out, and combining unrelated data. Planning, data collecting, and preparation are the best ways to prevent missing values. Some missing values are frequent and unavoidable . The challenge affects all data-related fields, including machine learning, statistics, and data-driven control.

Performance deterioration, data analysis challenges, and skewed results from missing and full value. The importance of missing values depends on how much data is missing, its pattern, and its process. Missing values may be managed by deleting instances and replacing them with prospective or projected values. Imputation is a more sophisticated approach to missing values. It may generate calculation and analysis errors and systematic data disparities. The data cleaning is explained in Algorithm 2.1 [30].

Algorithm (2.1) Data Cleaning

Input: *Two-dimensional array DS[n*m] where m is a matrix of input features and n is a vector of binary labels*

Output*: Two-dimensional array DB[n*m] after cleaning the data.*

```
for j = 1 to m
  if all values v in feature j are equal then
      ignore feature j
  end if
end for j
```

- **Outlier**

Statistical literature defines an outlier as an observation that is incongruous with other data. Outlier detection approaches strive to overcome the issue of "appears inconsistent," according to the authors. Statistical and probabilistic knowledge, distance and similarity-dissimilarity functions, metrics and kernels, accuracy with labelled data, association rules, pattern qualities, and other domain aspects were used to solve the challenge .

2.4.2 Encoding of Dataset

To facilitate its fitting to a machine learning model, categorical variables are often encoded into numeric values using an encoding approach.

- **Label encoding**

  Working with datasets that have overlapping labels in rows and columns is a typical technique in the field of machine learning. Identifiers might be words or numbers. The training data is frequently annotated with words in order to make it easier for humans to interpret. The act of converting labels from their human-readable form into a numerical representation that computers can process is known as label encoding. As a result, this might direct machine learning algorithms to use the labels more efficiently. A key first step is to get the structured dataset ready for supervised learning [30].

- **One-Hot encoding**

  The integer encoding may not be sufficient for categorical variables without an ordinal relationship and may potentially lead to model misinterpretation. Letting the model presume a natural ordering across categories rather than demanding an ordinal relationship through ordinal encoding (predictions halfway between categories) may lead to subpar performance or surprising outcomes. The ordinal representation in this scenario can be represented using a one-hot encoding approach. In this situation, the variable's integer-encoded representation is discarded, and a new binary variable is added for each distinct integer value. If the data is encoded as an integer for categorical variables without an ordinal connection, the model can be deceived. using an ordinal encoding to forcibly create a hierarchical order [30].

2.4.3 Normalize the Dataset

Normalization is either a mapping approach, a scaling method, or an initial processing step. Where we may extend the known range by using the already-existing one. It's useful for making forecasts and predictions. Numerous methods exist for making predictions and forecasts, but they don't always agree with one another[31]. Therefore, the Normalization method is necessary to bring the significant variance of prediction and forecasting closer together. Normalization aims to ensure that all features are in the same unit of measurement. Therefore, it is used to avoid the difference between the influence of small values and large values that dominate the results. Many methods are found for data normalization such as min-max, and z-score normalization. The Normalization is explained in Algorithm 2.2 [32].

- **Min-Max**

Min-Max normalization is a straightforward method that allows one to precisely fit data inside a set boundary. Equation 2.1 represents the Min-Max method of normalizing.

$$x'_{i,n} = \frac{x_{i,n} - \min(x_i)}{\max(x_i) - \min(x_i)} (nMax - nMin) + nMin \qquad (2.1)$$

Where X is the number of features, Min-Max normalization is a method for obtaining more meaningful information from chaotic data by using statistical measures such as the mean and standard deviation[32].

- **Z-score**

A Z-score normalization parameter is used for unstructured data and can be normalized using the z-Score parameter [32]. Equation 2.2 represents the Z-score normalization.

$$\bar{x}_{i,n} = \frac{x_{i,n} - \mu_i}{\sigma_i} \qquad (2.2)$$

Algorithm (2.2) Normalization

Input: *Two-dimensional array DS[n*m] where m is the number of features and n is a vector of binary labels*

Output*: Two-dimensional array DB[n*m] after applying normalization.*

```
for i = 1 to m
    set max and min to the first value of feature i
        for j = 1 to n
            if DS [j][i] < min then
                min = DS[i][j]
            else
                if DS [j][i] > max then
                    max = DS [j][i]
                end if
            end if
        end for j
    for j = 1 to n
            modify each value DS [j][i] in feature i according to the equation:
                $DS[j][i] = (DS[j][i] - min)/ max - min$
    end for j
end for i
```

2.5 Feature selection (FS)

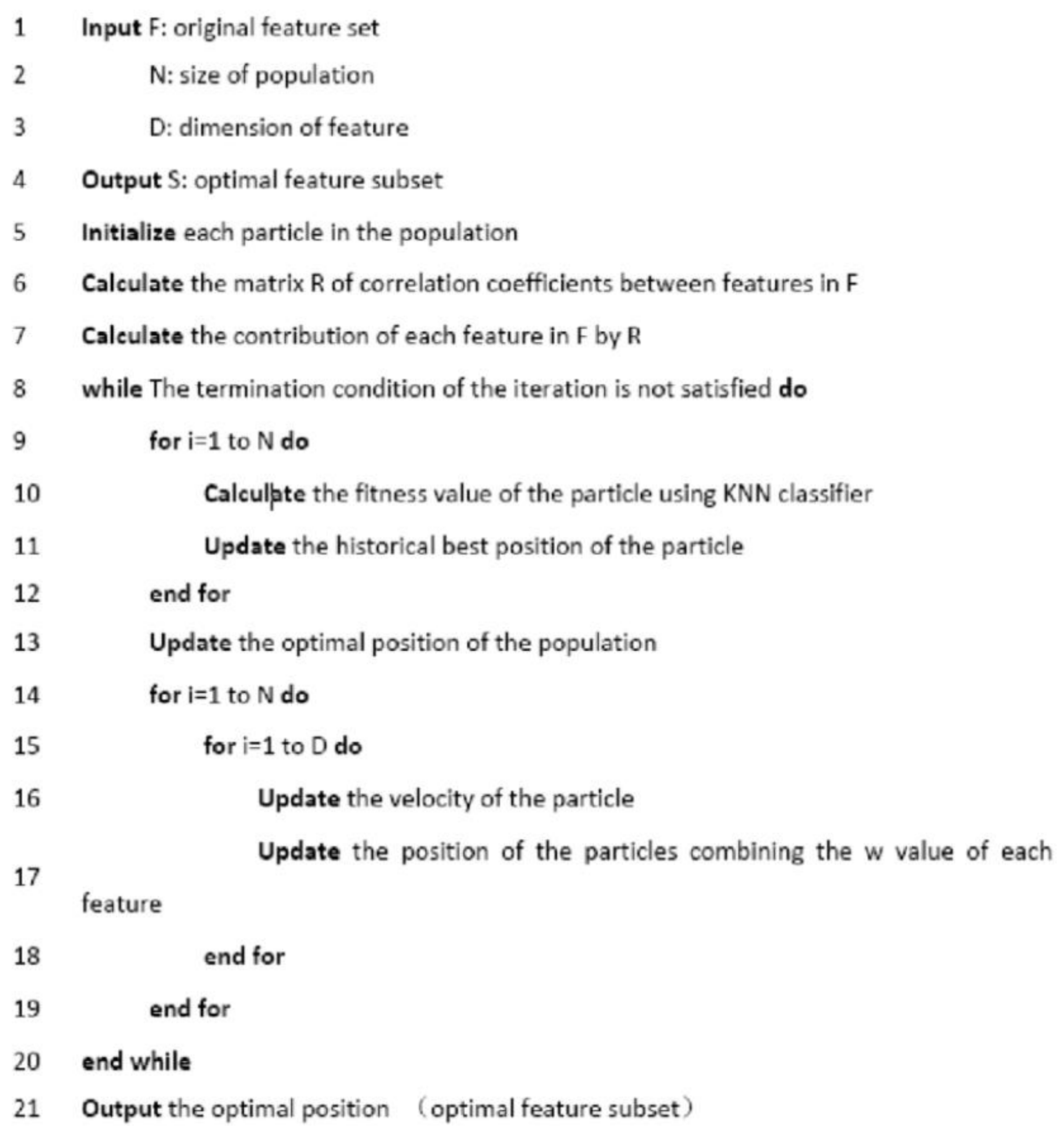
Feature selection (FS) is the process of choosing the most relevant and most affect set of features on the problem from the original features. Feature selection helps in reducing computation requirements, understanding data, improving the predictor performance, and reducing the effect of the curse of dimensionality[33]. Generally, features are classified into strongly relevant, weakly relevant, or irrelevant. The role of FS techniques is identifying the strongly relevant features of the target class. These selected features are sufficient to describe the target class and the deleted features do not have any positive effect on the performance of the prediction model. Thus, the accuracy of the model may be increased and the complexity of the model will be decreased accordingly[34].

- **Correlation Feature selection**

In machine learning, feature selection is a preprocessing procedure that can streamline results by reducing their dimensionality, eliminating undesired noise, boosting learning process dependability, and other factors. We refer to a feature as outstanding when it enhances the class as a whole without duplicating the work of other features. In other words, a feature's correlation with the class should be high but its association with other class traits should be low. Here, the main technique has been information theory based on entropy. A random variable's entropy measures how unpredictable it is [35]. Equation 2.3 represents the Entropy. The Correlation Feature selection is explained in Algorithm 2.3. Where X is the number of features.

$$H(X) = -\sum P\ (X_i)\ log2\ (P\ (X_i)) \quad (2.3)$$

Algorithm (2.3) Correlation Feature Selection



```
Input F: original feature set
      N: size of population
      D: dimension of feature
Output S: optimal feature subset
Initialize each particle in the population
Calculate the matrix R of correlation coefficients between features in F
Calculate the contribution of each feature in F by R
while The termination condition of the iteration is not satisfied do
    for i=1 to N do
        Calculate the fitness value of the particle using KNN classifier
        Update the historical best position of the particle
    end for
    Update the optimal position of the population
    for i=1 to N do
        for i=1 to D do
            Update the velocity of the particle
            Update the position of the particles combining the w value of each feature
        end for
    end for
end while
Output the optimal position （optimal feature subset）
```

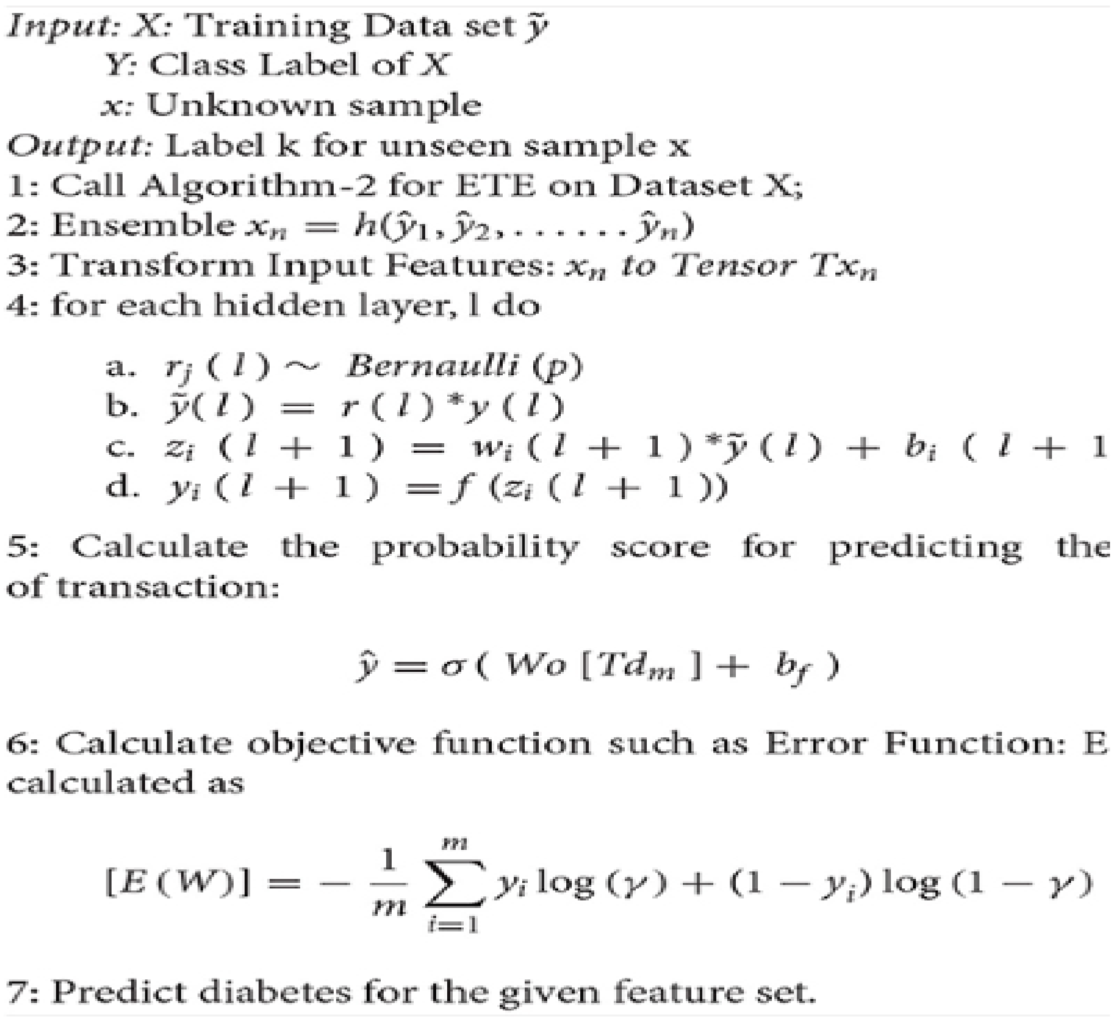
- **Extra tree feature selection**

The Extra-Trees feature selection uses a traditional top-down methodology to produce a large number of unpruned decision trees. For each node you want to divide, you must vigorously randomize the characteristics and cut points you use. In the worst situation, it creates trees whose topologies have nothing to do with the training output data. In particular, it deviates from standard tree-based ensemble methods in two significant ways [35]:

first, it uses the whole training sample to construct its trees (instead of a bootstrap replica), and second, it splits nodes by arbitrarily selecting their boundaries. The final prediction is decided by a majority vote based on the average predictions of all the trees. The idea behind the extra-trees classifier is that completely randomizing the cut-point and attribute, in tandem with ensemble averaging, would minimize variance more effectively than the weaker randomization methodology applied by other techniques. Instead than depending on bootstrap replicates. The algorithm's strength lies in its ability to do calculations quickly and with little wasted time. Like the previous methods, the Extra trees approach has found a broad range of applications in the academic[36].

After studying both feature selection types, in which the Extra Trees is noticeably quicker. Since a consequence, Extra Trees looks like an excellent option to employ if you're on the fence about which of the two ensembles to use, as the identical effect may be produced in a more expedient manner. The Extra tree feature selection is explained in Algorithm 2.4.

Algorithm (2.4) Extra tree Feature Selection



```
Input: X: Training Data set $\tilde{y}$
       Y: Class Label of X
       x: Unknown sample
Output: Label k for unseen sample x
1: Call Algorithm-2 for ETE on Dataset X;
2: Ensemble $x_n = h(\hat{y}_1, \hat{y}_2, \ldots\ldots \hat{y}_n)$
3: Transform Input Features: $x_n$ to Tensor $Tx_n$
4: for each hidden layer, l do
       a. $r_j(l) \sim Bernaulli(p)$
       b. $\tilde{y}(l) = r(l) * y(l)$
       c. $z_i(l+1) = w_i(l+1) * \tilde{y}(l) + b_i(l+1$
       d. $y_i(l+1) = f(z_i(l+1))$
5: Calculate the probability score for predicting the
of transaction:
       $\hat{y} = \sigma(Wo[Td_m] + b_f)$
6: Calculate objective function such as Error Function: E
calculated as
       $[E(W)] = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\gamma) + (1 - y_i)\log(1 - \gamma)$
7: Predict diabetes for the given feature set.
```

2.6 Classification Algorithms

Classification is a method of Machine Learning that use previously labeled data to discover where new data fits in a predefined taxonomy. Using preexisting data, this method employs a collection of algorithms to identify how a fresh batch of records should be organized into preexisting categories.

2.6.1 Support vector machine (SVM)

Support vector machines (SVMs) are regarded as one of the most reliable techniques for discriminative categorization. The Structural Risk Minimization notion from computational learning theory serves as a motivation for the SVM classifier[30]. This hypothesis aims to identify a testable presumption that will produce the least amount of "true mistake." Additionally, SVMs are easy to research and assess due to their strong theoretical foundation [37].

SVM uses both a positive and a negative training set, in contrast to the majority of other classification algorithms. The SVM won't be able to find the hyper plane that divides positive and negative data in n dimensions without a positive and negative training set. Document representations that are closest to the decision surface are referred to as "support vectors". The training set performance of SVM classification is unaffected by the removal of non-support vector documents. The classification performance of SVM is unparalleled. It manages texts with many dimensions and filters out unnecessary details[37].

Additionally, since similarity is determined differently for each category, classification of documents might lead to confusion. Finding the linear separating hyperplane that maximizes the margin between two data sets is the goal of the supervised learning method for classification known as SVM.

To calculate the margin, two parallel hyperplanes are "pushed up against" the two sets of data. Intuitively, the separation achieved by the hyperplane with the greatest distance from the nearest data points of both classes is the best separation, as the classifier's generalization error decreases with increasing margin[38]. The SVM algorithm is explained in Algorithm 2.5.

Algorithm (2.5) SVM algorithm steps

Input*: training data (X, y) where X is a matrix of input features and y is a vector of binary labels*

Output*: a linear classifier represented by the weight vector w and bias term b*

```
Begin

  Step1: Initialize weight vector w and bias term b to random values:

          w = random_initialization()

          b = random_initialization()

  Step2:  Define the cost function:

           def cost_function(X, y, w, b)

  Step3: Compute the margin for each sample:

          margins = y * (X.dot(w) + b)

  Step4:  Compute the cost as the sum of the hinge loss for samples with margin
  less than 1:
        cost = sum(max(0, 1 - margins[i]) for i in range(len(y)))
Step5:  Add a regularization term to the cost:
        cost += 0.5 * w.dot(w)
        return cost
Step6:  Define the gradient of the cost function:
        def grad_cost_function(X, y, w, b):
Step7:  Initialize gradients for w and b:
        grad_w = np.zeros(w.shape)
        grad_b = 0
        for i in range(len(y)):
Step8: If the margin is less than 1, update the gradients:
       if y[i] * (X[i].dot(w) + b) < 1:
       grad_w += -y[i] * X[i]
       grad_b += -y[i]
Step9: Add the gradient of the regularization term to the gradients:
       grad_w += w
```

```
        return grad_w, grad_b
Step10:  Train the model using gradient descent:
            for i in range(num_iterations):
  Step11:  Compute the gradients
            grad_w, grad_b = grad_cost_function(X, y, w, b)
  Step12: Update the parameters:
            w -= learning_rate * grad_w
            b -= learning_rate * grad_b
  Step13:  Print the cost at each iteration:
            print("Cost at iteration", i, ":", cost_function(X, y, w, b))
Step14: Return the weight vector w and bias term b:
        return w, b
End
```

2.6.2 Feed Forward neural network (FFNN)

A artificial feedforward neural network lacks a feedback loop since nodes are connected only one way. As a result, it differs from recurrent neural networks, which are its child. The first and most fundamental type of artificial neural network is the feedforward network. Feed-forward neural networks are the favored paradigm in many real-world applications.

They have been given various names. One illustration is "multilayer perceptron's" (MLP).The feed-forward neural network is a categorization method used in artificial intelligence that draws its inspiration from biology. Basic neuron-like processing units are coupled with each other and with all of the units in the layer below in layers upon layers. Each of these bonds may differ in terms of weight or strength. Connection weights are the form in which the information in a network is stored. A neural network's fundamental building pieces are called nodes[26]. The FFNN algorithm is explained in Algorithm 2.6.

Algorithm (2.6) FFNN algorithm steps

```
Begin
  Step1: Initialize weights and biases:
         weight1 = random_weights(20, 12)
         bias1 = random_bias(12)
```

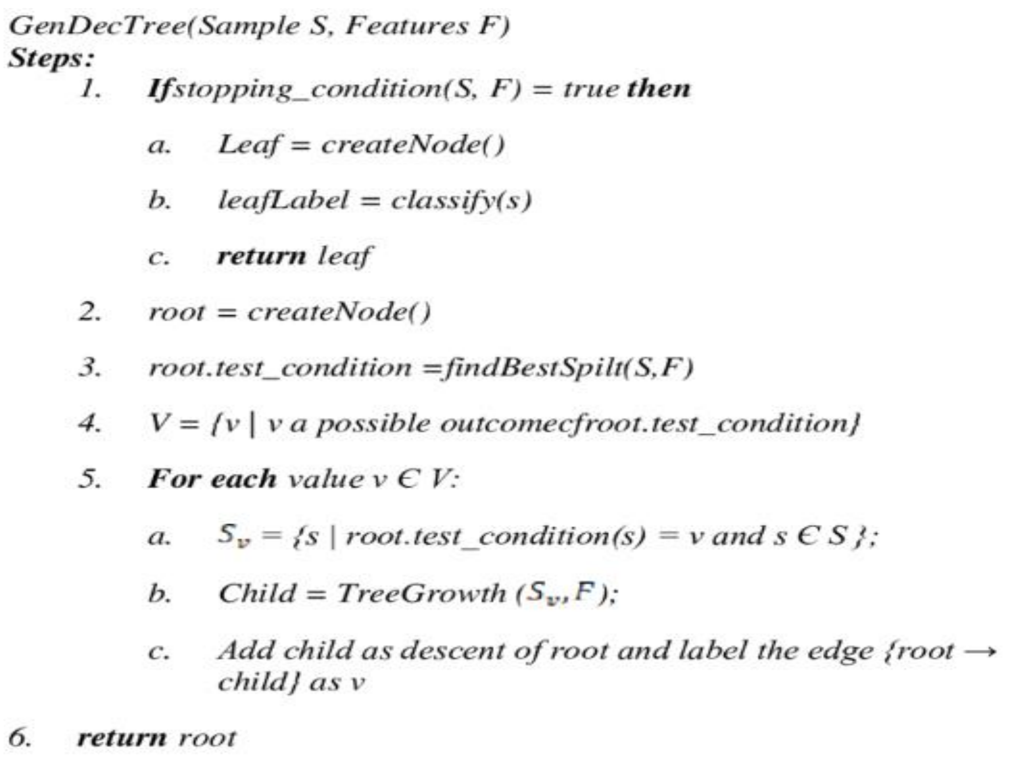
```
        weight2 = random_weights(12, 12)
        bias2 = random_bias(12)
        weight3 = random_weights(12, 2)
        bias3 = random_bias(2)
Step2: Define the Adam optimizer:
      optimizer = Adam(learning_rate=0.001)
      for each training example, x:
Step3:  forward pass through the network:
       input_layer = x
       hidden_layer1 = relu(dot(input_layer, weight1) + bias1)
       hidden_layer2 = relu(dot(hidden_layer1, weight2) + bias2)
       output_layer = softmax(dot(hidden_layer2, weight3) + bias3)
Step4: calculate the binary cross-entropy loss:
       loss = binary_crossentropy(y, output_layer)
Step5:  backward pass to update weights and biases:
      grad_weight3,      grad_bias3      =      backprop(output_layer,
hidden_layer2, y, loss, weight3)

        grad_weight2, grad_bias2 = backprop(hidden_layer2, hidden_layer1, y, loss, weight2)

        grad_weight1, grad_bias1 = backprop(hidden_layer1, input_layer, y, loss, weight1)

  Step6:  update weights and biases using the Adam optimizer:

        weight3 = optimizer.update(weight3, grad_weight3)

        bias3 = optimizer.update(bias3, grad_bias3)

        weight2 = optimizer.update(weight2, grad_weight2)

        bias2 = optimizer.update(bias2, grad_bias2)

        weight1 = optimizer.update(weight1, grad_weight1)

        bias1 = optimizer.update(bias1, grad_bias1)

End
```

2.6.3 Decision Tree (DT)

Classifier-generating systems are a common data mining strategy. Categorization algorithms are used in data mining because of their capacity to handle big datasets. It can be used to categorize newly available data as well as to classify knowledge based on training sets and class labels. This work focuses on the decision tree algorithm, one of the numerous categorization methods available in machine learning[33].

DT is a well-liked categorizing technique. Nodes and branches make up each tree. The nodes indicate the attributes of a class that needs to be categorized, and the subsets describe the possible values for each node. Because of how simple it is to analyze them and the great degree of accuracy they may achieve across a number of data types, decision trees have been frequently used[34]. The DT algorithm is explained in Algorithm 2.7.

Algorithm (2.7) Decision Tree (DT) algorithm steps



```
GenDecTree(Sample S, Features F)
Steps:
    1.  **If**stopping_condition(S, F) = true **then**
        a.  Leaf = createNode()
        b.  leafLabel = classify(s)
        c.  **return** leaf
    2.  root = createNode()
    3.  root.test_condition =findBestSpilt(S,F)
    4.  V = {v | v a possible outcomecfroot.test_condition}
    5.  **For each** value v ∈ V:
        a.  $S_v$ = {s | root.test_condition(s) = v and s ∈ S };
        b.  Child = TreeGrowth ($S_v$,F);
        c.  Add child as descent of root and label the edge {root →
            child} as v
6.  **return** root
```

2.7 Voting

In the voting method, each malware sample is subjected to independent analysis by several classifiers, with the results then being combined to determine a final classification. feel that this strategy has a great deal of potential for enhancing the efficiency of malware detection because it has been demonstrated to improve the accuracy and resilience of machine learning algorithms in a range of applications[36].

The voting method analyzes each malware sample using a variety of classifiers, each with advantages and disadvantages. The predictions from each classifier are merged in some fashion to get the final prediction for the sample's class label. The forecasts can be combined in a variety of ways, such as majority voting, weighted voting, and stacking.

The voting method can reduce the possibility of false positives and false negatives in machine learning algorithms, which is one of its main advantages. It is possible to lessen the effects of any given classifier that can produce unreliable results by using numerous classifiers with various strengths and weaknesses. In the context of malware detection, where accuracy and dependability are crucial[37], this can help to increase the algorithm's overall accuracy and resilience. Voting classifier supports two types of voting:

2.7.1 Hard Voting

The class with the highest majority of votes, or the class with the highest probability of being predicted by each of the classifiers, is the expected output class in the hard vote. In this case, the majority expected A to be the output when three classifiers (Malware (1), Malware (1), and Normal (0)) predicted the output class). Thus, the final prediction would be Malware (1) [38]. This is the type that was used in the proposed system because it works on the binary system (0,1).

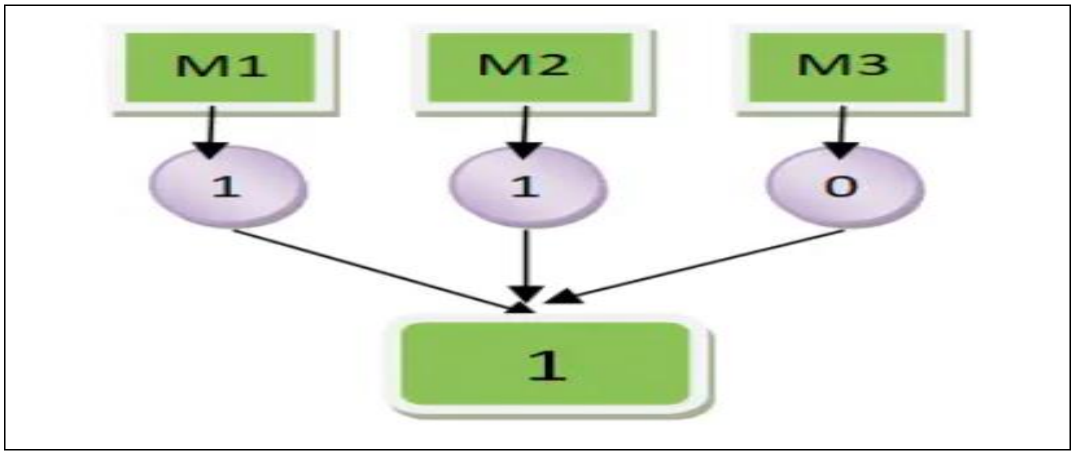


Figure 2.2 Hard Voting[38]

2.7.2 Soft Voting

With a soft vote, the forecast for the output class is based on the likelihood assigned to that class on average. Assume that given some input, the prediction probabilities for classes (Malware (1) and Normal (0)) are (0.30, 0.47, and 0.53) and (0.20, 0.32, 0.40). Because class (1) has the highest probability as averaged by each classifier, its average is 0.4333, whereas class (2) average is 0.3067, making class (1) the winner[39].

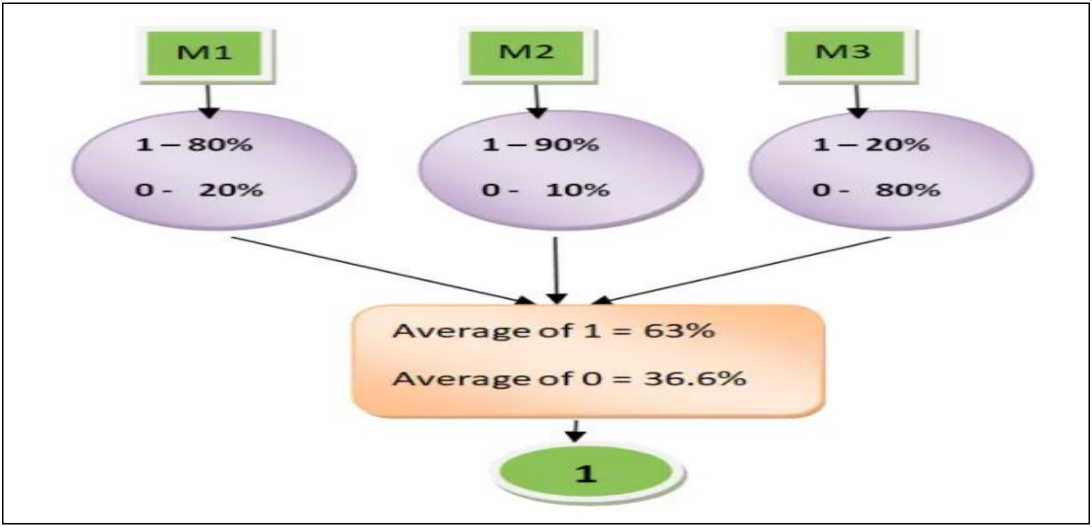


Figure 2.3 Soft Voting[39]

2.8 Evaluation (Confusion Matrix)

The quality of a statistical or machine learning model is measured using evaluation metrics. It is vital to evaluate machine learning models and algorithms for every project. There are several sorts of assessment metrics that may be used to evaluate a model. The following criteria are used to assess the efficacy of the ensemble machine learning models: The area under the receiver operating characteristics curve [40].

- Accuracy
- Precision
- Recall
- F1-score

The confusion matrix is used to measure the type of errors produced by a classifier. As presented in Figure 2.4, the component of a confusion matrix is True Positive (TR), True Negative (TN), False Positive (FP) and False Negative (FN). Metrics of accuracy performance might be significant when dealing with unbalanced data. The Confusion matrix and its accompanying words, which are deceptively simple despite their complicated appearance. The confusion matrix, precision, recall, and F1 score provide a more intuitive understanding of prediction outcomes than accuracy[41].

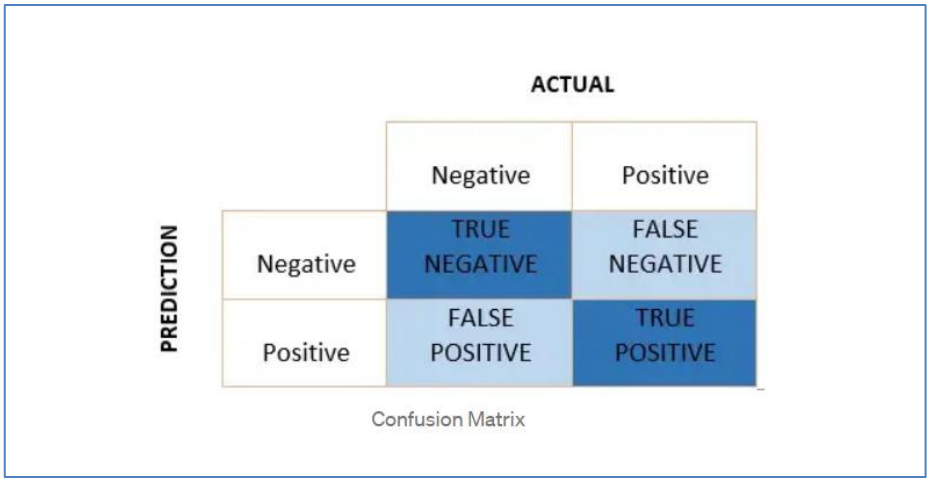


Figure 2.4 Confusion matrix[41]

When assessing the efficacy of a classification model, one uses a Confusion matrix, which is an N-by-N matrix where N is the number of target classes. In this matrix, can see how well the machine learning model did in predicting the actual target values. This provides a comprehensive analysis of our categorization model's accuracy and error rates[42].

1. **True Positive (TP)** — model correctly predicts the positive class (Positive means Normal).
2. **True Negative (TN)** — model correctly predicts the negative class(Negative means Malware).

3. **False Positive (FP)** — model gives the wrong prediction of the negative class (predicted-positive, actual-negative).
4. **False Negative (FN)** — model wrongly predicts the positive class (predicted-negative, actual-positive).

These standards are the standard for evaluating ML model performance. Their respective definitions follow:

Accuracy: the most widely used—and hence most abused—scoring method. This method works in some categorization problems. By dividing the total number of predictions by the number of correct predictions (including both classification and non-classification), accuracy is calculated. In the context of the current scenario, it is equally valuable to correctly anticipate whether or not an observation belongs to a specific class and whether or not it does[42].

Although it is debatable whether or not categorization challenges of this nature do exist, it is not generally advised to utilize them as a performance measure. The frequency with which a classifier properly labels occurrences is known as accuracy. The ratio of true positives and true negatives to all positive and negative observations is the definition of model accuracy, a performance criterion for machine learning classification models. In other words, accuracy is the frequency with which we expect our machine learning model to correctly predict an event, compared to the total number of predictions it has made[43]. Equation 2.4 represents the Accuracy.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \qquad (2.4)$$

Precision refers to how well a classifier is able to anticipate positive examples, defined as the fraction of cases where the classifier's prediction was correct. In statistics, precision is measured as the proportion of right predictions relative to the total number of predictions. When a model has a low error in separating out data points that do not belong to a given class, it is said to have high precision.

The percentage of successfully predicted positive labels is measured by the model accuracy score. The predictive value of the positive is another name for precision. Recall and precision are combined to balance false positives and false negatives. The distribution of classes has an impact on accuracy. Accuracy will decline as there are more samples drawn from the minority group. One could interpret precision as a sign of accuracy or excellence. We would pick a model with a high level of accuracy to lessen false negatives[43].

A model with a high recall would be the one we would use if we wanted to lessen false positives. When accuracy is crucial, as it is in medical diagnosis and spam filtering, false positives cost more than false negatives. The accuracy score is a crucial sign of a successful forecast when classes are severely imbalanced. It displays the quantitative relationship between true positives and the sum of true positives and false positives. According to the formula, the accuracy score is influenced by the value of false-positives. Therefore, if a high accuracy score is crucial for achieving business goals, you can decide to give priority to the creation of predictive models that produce fewer false positives [43]. Equation 2.5 represents the Precision.

$$precision = \frac{tp}{tp + fp} \qquad (2.5)$$

Recall: the fraction of correctly anticipated positive cases (out of all positive instances). The percentage of a class's members whose labels were properly predicted is known as recall. The model provides details on how many genuine members of a given class end up being misclassified or assigned to another category. The score for model recall reflects the model's ability to properly forecast the positives from the actual positives. This is in contrast to precision, which assesses how many positive predictions generated by models are truly accurate [44] .

A high recall score indicates that the model is proficient at identifying successful cases. On the other side, a low recall score indicates that the model is ineffective at identifying affirmative cases. In order to provide a complete picture of the model's performance, recall is sometimes paired with other performance measures, such as precision and accuracy. It shows the mathematical relationship between actual positives and the sum of real positives and real negatives. The computation that follows shows how the recall score is impacted by the value of false-negatives. Therefore, if a high recall score is crucial for achieving business goals, you can decide to give priority to the creation of predictive models that have fewer false negatives[44]. Equation 2.6 represents the Recall.

$$recall = \frac{tp}{tp + fn} \qquad (2.6)$$

Model **F1 score** reflects the model score as a function of the recall and accuracy scores. It is an alternative to Accuracy metrics (it does not need us to know the entire number of observations). It is often used as a single number that offers high-level information on the output quality of the model. This is a valuable model performance metric in situations when one attempts to maximize either the accuracy or recall score at the expense of model performance[44] .

The f1 score indicates the harmonic mean of accuracy and memory. The F1 Score is another metric that at first look seems contradictory. False Positives and False Negatives are two types of judgment errors that are combined into a single measure. The phrase alludes to the mathematical compromise between recall and accuracy. All of the truly encouraging observations for a certain class ought to be included, and those that don't belong there should be carefully left out. If we were to be successful, we would have superb precision and memory. This will thus ensure a high F1-Score consistent with our model[45]. Equation 2.7 represents the F1- Score.

$$f1_{score} = \frac{2 \times precision \ \times recall}{precision + recall} \qquad (2.7)$$

Chapter Three
The Proposed Model

3.1 Overview

In this chapter, the main stages of the system proposed in this thesis will be explained. First, the architecture of the proposed system is presented. Then follows a description of the data set and the pre-processing steps that were used. A description of the feature selection methods applied and an explanation of the classification techniques that are implemented.

3.2 Proposed model Architecture

The proposed model includes seven stages. It consists of several steps to achieve the objectives of the thesis as shown in Figure 3.1.

The first stage includes the different datasets (DDoS 2019, Big 2015, Malware Analysis). The second stage is preprocessing and this stage contains different steps to clean and prepare data for the next stage. The third stage selects feature from data using correlation and extra tree methods. The fourth stage involves dividing data into training data to train models and testing to evaluate the models. The fifth stage is a classification that applied various machine learning techniques to identify if the data is malware or not. The sixth stage is the vote on the classifier with the best accuracy. The seventh stage is an evaluation to evaluate the performance of the model.

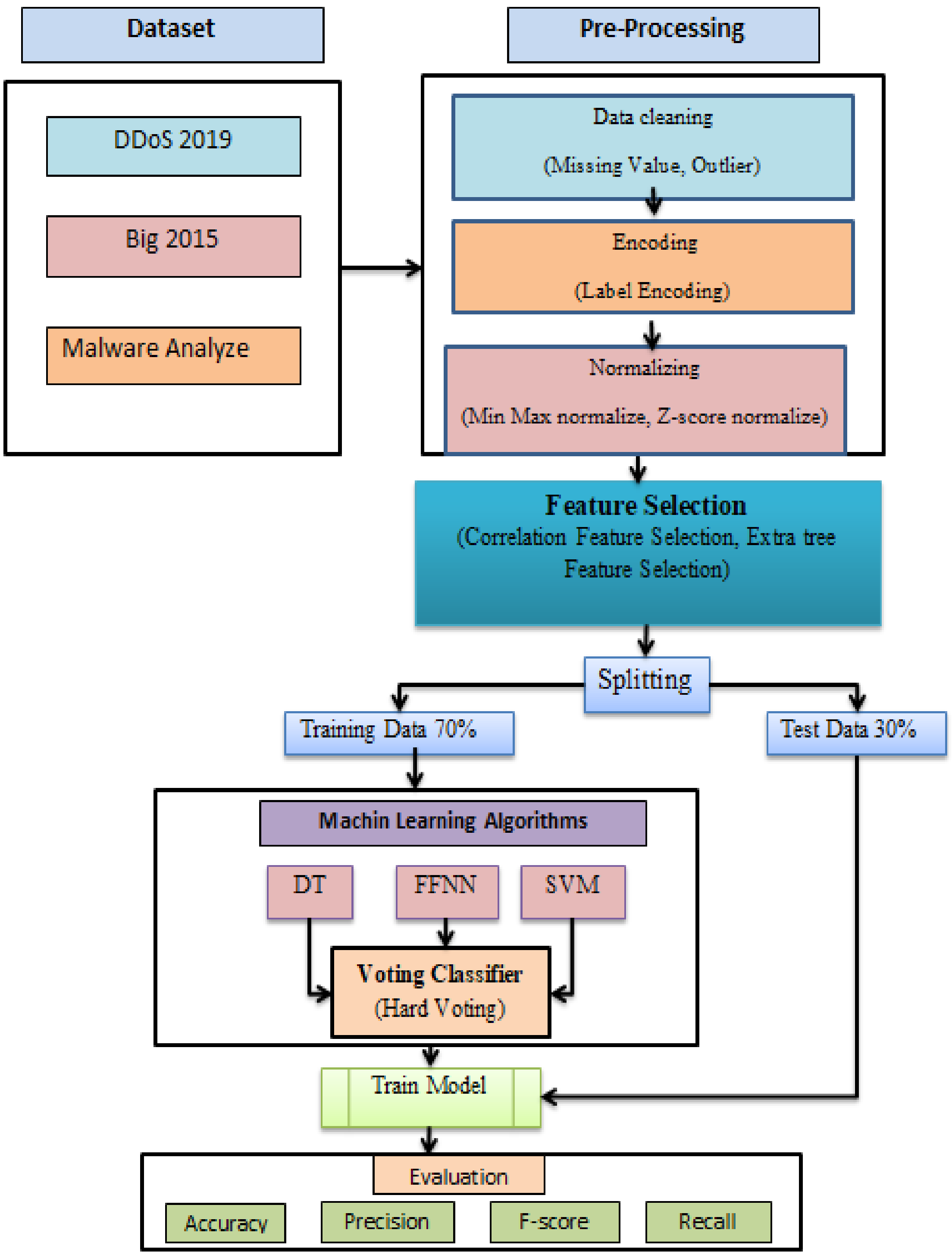


Figure 3.1 The Proposed Model Diagram

3.2.1 Dataset

Without input data, a machine learning system is impossible to create. Because of their data-intensive nature, machine learning models often need a significant amount of input before yielding accurate results. Even if you've built fantastic algorithms for machine learning models, the quality of your data is still crucial. This remark is the most concise and accurate explanation of how a machine-learning model works. One of the most crucial and time-consuming parts of any machine learning project is getting the data ready for analysis.

Log data for several types of malware may be discovered among the collection's multiple files. Various models may be trained using the retrieved log characteristics. There are three utilized datasets have many families of malware identified in the samples. The next section explains the datasets:

- **DDoS 2019 dataset**

CICDDoS 2019 offers PCAP files for both benign and current DDoS assaults. CICDDoS 2019 includes a training dataset and a testing dataset. The twelve DDoS assaults included in the training dataset are NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDPLag, WebDDoS, SYN, and TFTP. The testing dataset contains seven different forms of DDoS attacks: PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN . The samples of the DDoS dataset as presented in Table 3.1.

Table 3.1 The samples of the DDoS dataset

Sample	**Number of Labels**
BENIGN	31616
DrDoS_NTP	14365
DrDoS_DNS	3402
UDP-lag	3266
DrDoS_UDP	2157
DrDoS_MSSQL	2006
DrDoS_NetBIOS	1707
DrDoS_LDAP	1612
DrDoS_SNMP	1507
DrDoS_SSDP	763

- **Big2015 dataset**

Microsoft's Malware Classification Challenge dataset Big 2015. Anti-malware researchers have divided this dataset into nine categories. Since the competition has already concluded, are unable to submit the results and find out how well we did since the testing dataset does not include ground truth. The dataset has 300,000 records divided into normal/attack. The samples of the Big 2015 dataset as presented in Table 3.2.

Table 3.2 The dataset have 300,000 record divided into normal/attack

	Data Record	**Label**
Attack	133633	0
Normal	166367	1

- **Malware analysis dataset**

A malware analysis dataset is a collection of records that contain information related to malware samples. Each record typically includes a variety of data related to the malware sample, such as its behavior, characteristics, and attributes. The data in each record can be used to identify and analyze the malware, understand its potential impact, and develop strategies to prevent or mitigate its effects.

In addition to records containing information about individual malware samples, a malware analysis database may also include other types of data and information. For example, it may contain metadata about the samples, such as the date and time they were first discovered and analyzed, and the researcher or organization that performed the analysis.

The dataset is well-balanced since it is comprised of equally proportional amounts of both malicious and benign memory dumps. The following table provides a classification breakdown for several families of malware. The collection includes a total of 58,596 entries, of which 29,298 are safe to use and 29,298 are unsafe to use. The following table provides an overview of the overall number of malware families that fall under each category. The samples of the malware families as presented in Table 3.3.

Table 3.3 malware families

Malware category	Malware families	Count
Trojan Horse	Zeus	195
	Emotet	196
	Refroso	200
	scar	200
	Reconyc	157
Spyware	180Solutions	200
	Coolwebsearch	200
	Gator	200
	Transponder	241
	TIBS	141
Ransomware	Conti	200
	MAZE	195
	Pysa	171
	Ako	200
	Shade	220

3.2.2 Pre-Processing

The gathered data is in the form of unprocessed binary code saved in the form of executable files. Preprocessed them so that they would be useful for purposes. To begin, use a virtual computer to unpack the executables (VM). To enable automated unpacking of executable archives. It is necessary to preprocess the dataset in order to make it appropriate for categorization.

These steps are crucial for enhancing classification model performance and preparing the data for usage with machine learning and deep learning algorithms. Overfitting may also be prevented with the use of data balancing procedures, which are often used in machine learning methods. Figure (3.1) depicts the suggested approach to this preprocessing work, which can be broken down into three distinct phases: removing missing and noisy values from the data, encoding the text values, and normalizing large numbers.

- **Data cleaning**

In this step, the three datasets used (DDoS 2019, Malware Analysis, Big 2015) have been cleaned and all spaces and noise data have been removed. As a first step in cleaning the data, removed unimportant columns for the model are building such as the ID number, Un name, and Appears, while the features that affect the prediction state and have important roles remain preserved such as the label, the source IP address, and the destination IP address.

- **Encoding Process**

Numerous strategies exist for turning a textual string into a numerical value. The most often used are Label Encoding and One Hot Encoding. The proposed system used Label Encoding to convert unique data to serial numbers.

- **Normalization Process**

Each dataset contains a large number of numbers. As a result, these numbers slow down and distort the calculations' outputs, as well as make them more difficult to apply. The normalization approach was created to deal with this problem (the requirement to reduce the size of large numbers) without compromising any of the information contained within. Min-Max and Z-Score normalization are two common ways to do this activity. Both methods were used in this thesis. This was performed for all values of numeric features that would be the input for the machine learning algorithms.

3.2.3 Splitting Data

- **Training Data**

  The information used to train the model would be stored in a train set. This information would be used to train model. When used to cost reduction and accurate data prediction, It takes 70% of the data.

- **Testing Data**

The data used to evaluate how well the trained and validated model performs under real-world conditions may be found in the test set. It informs us how frequently our model is to forecast something that does not make sense and how efficient it is as a whole. The model may be measured in a variety of ways, some of which include precision, recall, accuracy, and others. It takes 30% of the data. The data splitting for the three utilized datasets as presented in Table 3.4.

Table 3.4 data splitting for the three utilized datasets

Dataset Name	**Training**	**Testing**
DDoS2019	43875 samples	18804 samples
Malware Analyze	151445 samples	64906 samples
Big 2015	210000 samples	90000 samples

3.2.4 Machine learning classification algorithm

The categorization algorithms are composed of three levels: the input layer, the hidden layers, and the output layer. All incoming data must be initialized by the input layer. The experience from the previous phase is used to represent the input data during the model-training step. The testing data or brand-new, unlabeled data may be used as the input data during the evaluation process. Between the input layer and the output layer, the hidden layers act as a transitional layer and are where all computation—including feature extraction and classification—takes place. For each data input, the output layer generates the final results. Figure 3.2 depicts the neural network.

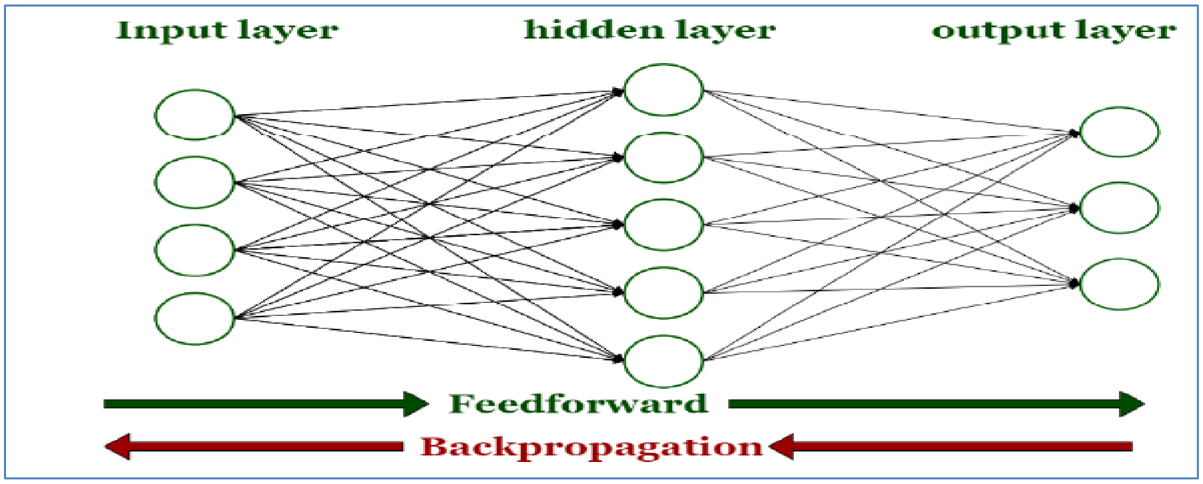


Figure 3.2 Neural Network

In a neural network, back propagation and feed forward propagation are the two types of propagation. As shown in Figure 3.2, Feed Forward Propagation is used to depict forwarding from the input layer to the hidden levels and then to the output layer. As shown in the equation below, this

Forwarding uses a perceptron classifier to predict outcomes and provide output results from input data. Equation 3.1 represents the Activation Functions.

$$y = \sum_{i=1}^{n} x_i \; w_i + b \qquad (3.1)$$

In this equation, x represents the information being processed by the neuron, w is the weight being applied to the neuron, and b represents the bias. The data obtained will then be included into the Activation Functions (more details will explain in the following subsection).

Instead of sending information from the output layer to the input layer, as in Feed Forward Propagation, information is sent in the reverse direction, i.e., backwards. The fundamental goals of Back Propagation are model learning through weight and bias adjustment. It uses optimizer and loss function methods. The following algorithms runs for the three datasets (Big2015, Malware analyzer, DDoS2015).

- FFNN

  The FFNN model with correlation feature selection and Z-Score normalization showed the best performance on the test data. However, this model is also the most computationally expensive, which may limit its practical use in certain scenarios.
- SVM

  The SVM model with DT feature selection and Min-Max normalization show good performance on both the training and test data. This model was also relatively computationally efficient, making it a good option for practical use.

- DT

  The DT model with extra tree feature selection and Z-Score normalization show the best performance on the test data.

3.2.5 Machine learning classification algorithm with Hard Voting

The use of dynamic machine learning algorithm selection for multiple malware detection shows promise in improving the accuracy of malware detection systems. However, the choice of algorithm and preprocessing techniques will depend on the specific needs and limitations of the system being used.

Based on the evaluation results provided, it is difficult to determine the best method for dynamically detecting multiple malware. To achieve dynamic machine learning algorithm selection for multiple malware detection, voting was used to Developing a dynamic selection classification system that utilizes a hard voting approach to select the best algorithm for detecting multiple types of malware based on their specific characteristics and behaviors.

Chapter Four
Result and Discussion

4.1 Introduction

This chapter introduces dynamic machine learning algorithms for multiple malware detection, which combine the power of dynamic analysis with the ability of machine learning to identify and classify malware. The chapter will discuss various techniques for dynamic analysis, including sandboxing, system call monitoring, and behavior analysis. It will also cover various machine learning algorithms, such as neural networks (NN), decision trees (DT), and support vector machines (SVM), that can be used to classify and identify different types of malware.

The chapter will provide a detailed analysis of the strengths and weaknesses of each dynamic analysis technique and machine learning algorithm. It will also discuss the challenges associated with dynamic machine learning algorithms for multiple malware detection, such as the need for large-scale data sets and the potential for false positives and false negatives.

4.2 Results of DDoS 2019 Dataset

The DDoS 2019 dataset is a publicly available dataset that contains network traffic data related to Distributed Denial of Service (DDoS) attacks. This dataset was compiled by the Network Security Research Lab at the University of New Brunswick, Canada, and was released in 2019.

The dataset contains a total of 12,895 network traffic captures, which were collected over a period of 30 days. These captures were taken from a real-world network environment, and include both DDoS attack traffic and normal network traffic. The captures were taken at a rate of one capture per minute, resulting in a total of 43,200 captures over the 30-day period.

Of the 12,895 captures in the dataset, 9,677 are labeled as normal traffic and 3,218 are labeled as DDoS attack traffic. This means that approximately 25% of the captures in the dataset represent DDoS attacks.

The DDoS attacks in the dataset are classified into three different categories: TCP Flood, UDP Flood, and HTTP Flood. TCP Flood attacks are the most common type of attack in the dataset, accounting for 76% of all DDoS attacks. UDP Flood attacks account for 23% of attacks, while HTTP Flood attacks make up just 1% of attacks.

The dataset includes various features and attributes for each network traffic capture, such as source and destination IP addresses, source and destination port numbers, protocol type, packet size, and time stamp. These features can be used to develop machine learning models that can accurately identify DDoS attacks and distinguish them from normal traffic.

4.3 Normalizing

4.3.1 Z-score Normalizing

The suggested method employed a min-max normalization technique where the minimum and maximum values are reset to 0 and 1, respectively. This normalization resulted in a range of outcomes between 0 and 1. The columns 'Flow ID', 'Source IP', 'Source Port', 'Destination IP', 'Destination Port', 'Protocol', 'Fwd Packet Length Max', 'Fwd Packet Length Min', and 'Fwd Packet' underwent normalized processing into z-score normalizing.

4.3.2 Min-Max Normalizing

The columns 'Inbound', 'Source IP', 'Flow ID', 'Destination IP', 'Min Packet Length', 'Protocol', 'Destination Port', 'URG Flag Count', 'Source Port', 'Fwd Packet Length Min', 'Avg Fwd Segment Size', 'Down/Up Ratio', 'min_seg_size_forward', 'Packet Length Mean', 'Fwd Packet Length Mean', 'CWE Flag Count', 'Flow Bytes/s', 'Bwd Packet Length Min', 'Bwd Packet Length Mean', 'Avg Bwd Segment Size', and 'Label' are processed using min-max normalization.

4.4 Label Encoding of Dataset

As stated earlier, a Label Encoder was employed to transform the textual information into binary values. The textual data that underwent this transformation in the dataset includes 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp', and 'label'.

4.5 Feature Selection

This study aims to predict malware and highlight the most important features that may affect performance, however, not all of these features are suitable for the prediction task and some of them reduced the accuracy of the prediction model and increased the time complexity. For this reason, the feature selection process has been used in this thesis.

4.5.1 DDoS2019 Dataset

- The first feature selection is the Extra Tree feature selection from the DDoS2019 dataset. The following figure 4.3 shows the selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

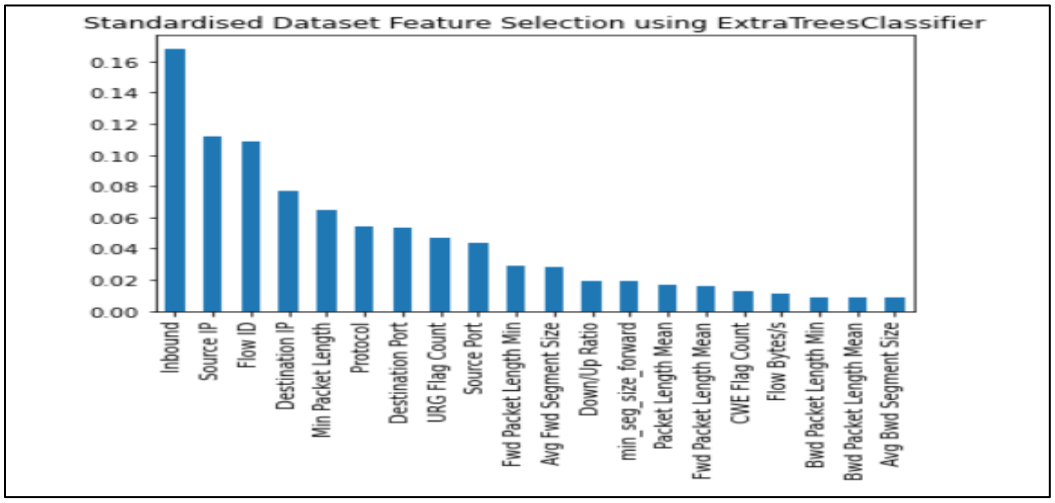


Figure 4.1 DDoS 2019 dataset feature selection using Extra Trees feature selection

- The second feature selection is the Correlation feature selection from the DDoS2019 dataset. The following figure 4.4 shows selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

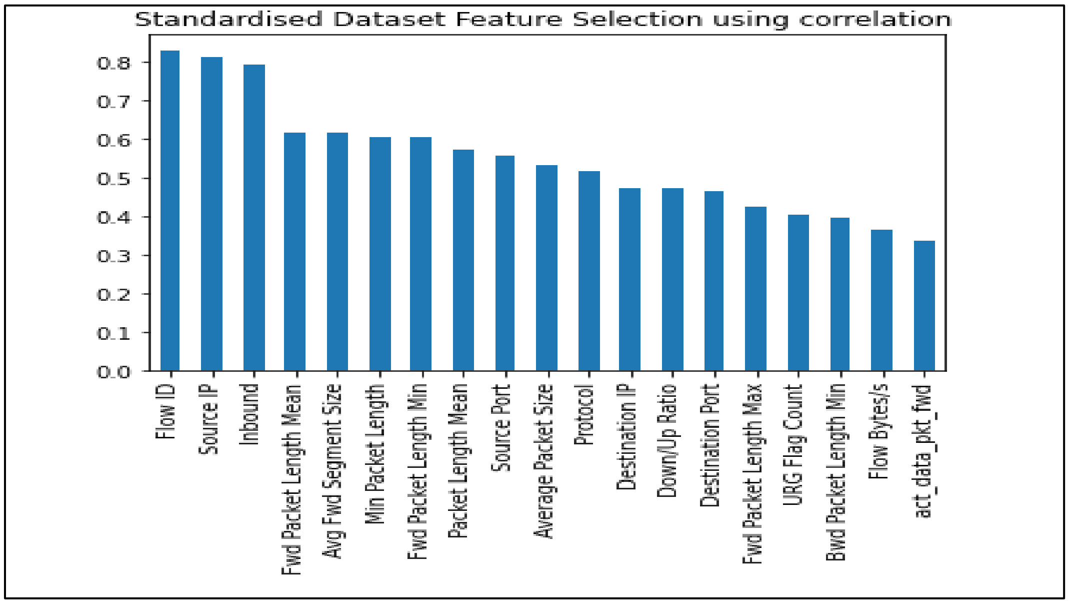


Figure 4.2 DDoS 2019 dataset feature selection using Correlation feature selection

4.5.2 Big2015 Dataset

- The first feature selection is the Extra Tree feature selection from the Big2015 Dataset. The following figure 4.5 shows selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

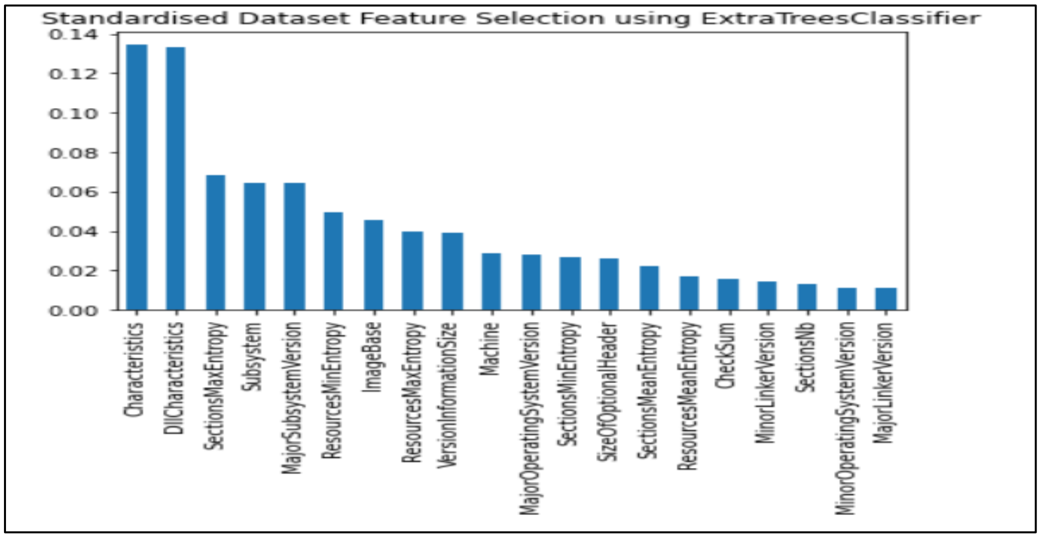


Figure 4.3 Big2015 dataset feature selection using Extra Trees feature selection

- The second feature selection is the Correlation feature selection from the Big2015 Dataset. The following figure 4.6 shows selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

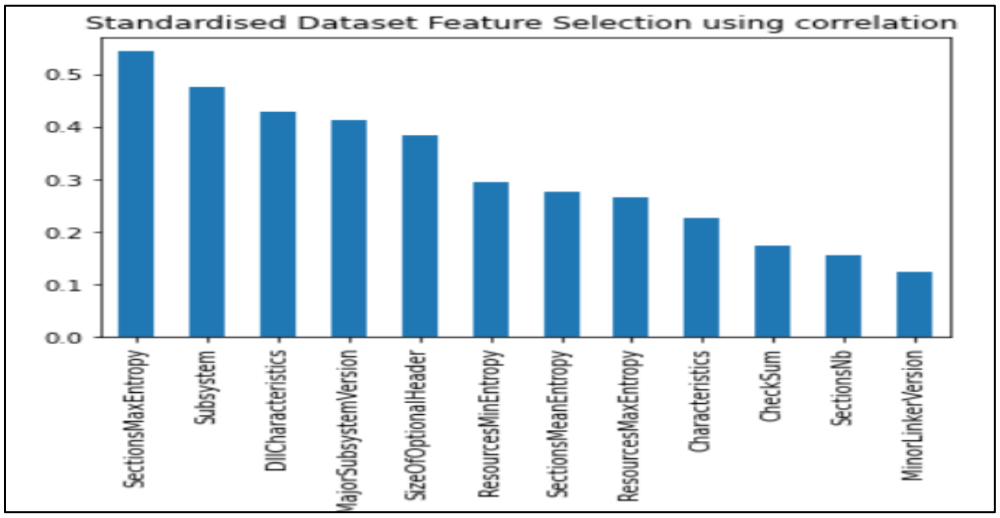


Figure 4.4 Big2015 dataset feature selection using Correlation feature selection

4.5.3 Malware Analysis Dataset

- The first feature selection is the Extra Tree feature selection from the Malware Analysis Dataset. The following figure 4.7 shows selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

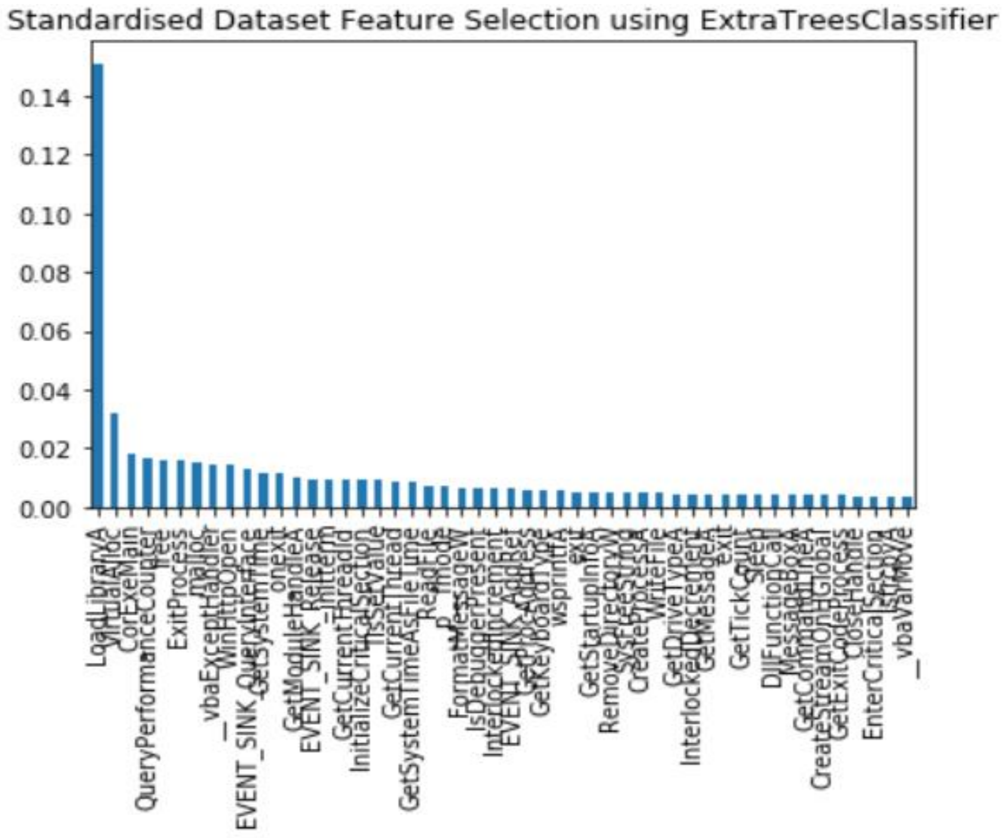


Figure 4.5 Malware analysis dataset feature selection using Extra Trees feature selection

- The second feature selection is the Correlation feature selection from the Malware Analysis Dataset. The following figure 4.8 shows selected features, where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection where the X-axis represents the percentage of closeness between the feature and label, the Y-axis is the number of feature selection.

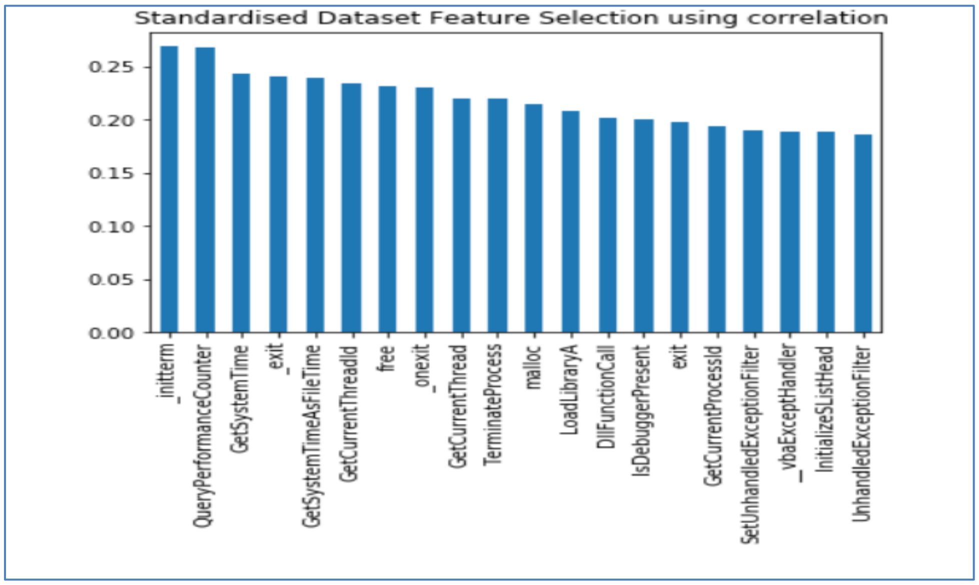


Figure 4.6 Malware analysis dataset Feature Selection using Correlation feature selection

4.6 Splitting Dataset

1. Split the dataset into two groups: training data and testing data.

- Training data: 70% of the original dataset.

- Testing data: 30% of the original dataset.

2. Within the training data, divide the instances into two groups based on their labels: normal and malware.

- Normal labels: 31,349 instances.
- Malware labels: 31,330 instances.

4.7 Classification

After applying preprocessing and feature extraction. The data became proper to classify by machine learning algorithms. Three types of machine learning algorithms were performed: FFNN, SVM, and DT.

4.7.1 Results of Big 2015 Dataset

The Big2015 dataset is a large collection of anonymized mobile phone usage data, which was released by researchers at MIT's Media Lab in 2016. This dataset contains call and text message records from over 1.5 million mobile phone users in Ivory Coast, Africa, over the course of a 6-month period in 2013.

The goal of the Big2015 dataset was to provide researchers with a unique dataset to study human behavior and social dynamics, particularly in the context of developing countries. By analyzing the data, researchers hoped to gain insights into a variety of areas, including patterns of communication, social interactions, and economic activity.

The dataset includes information such as the date and time of each call or text message, the duration of the call, and the location of the user at the time of the communication. This information was anonymized to protect the privacy of the users, but it still allowed researchers to identify patterns and trends in the data.

One of the most interesting findings from the Big2015 dataset was that mobile phone usage patterns were strongly correlated with economic indicators, such as GDP and poverty rates. For example, areas with higher levels of economic activity tended to have more frequent and longer phone calls, while areas with lower levels of economic activity tended to have more text messages.

Another key finding was that social networks tended to be strongly localized, with most communication occurring within small geographical areas. This suggests that social ties are closely linked to physical proximity, even in an age where communication can take place over long distances. The big data 2015 datasets as presented in Table 4.1.

Table 4.1 big data 2015 malware/normal number of records

Malware /Normal	Number of Instances
1(malware)	166367
0(normal)	133633

A. Results of Decision Tree

- Decision Tree model with ET feature selection and Z-score normalization on test data. The classification result for a Decision

Tree model with Extra Tree feature selection and Z-score normalization on test data as presented in Table 4.2.

Table 4.2 the classification result for a Decision Tree model with Extra Tree feature selection and Z-score normalization on test data.

Class	**Precision**	**recall**	**f1-score**	**support**
0(normal)	0.87	0.91	0.89	40165
1(malware)	0.92	0.89	0.91	49835
Accuracy			0.90	90000
macro avg	0.90	0.90	0.90	90000
weighted avg	0.90	0.90	0.90	90000

The table above shows the classification result for DT (Decision Tree) model applied to the testing data. The precision, recall and f1-score metrics are provided for each class (0 and 1) and for both the macro and weighted averages. The support column indicates the number of instances for each class in the testing set.

The DT model achieved a weighted average f1-score of 0.90, indicating that the model has a good balance of precision and recall for both classes. The precision for class 0 was 0.87, indicating that 87% of the instances predicted as 0 by the model were actually 0. The recall for class 0 was 0.91, indicating that the model correctly identified 91% of the instances that were actually 0.

The precision for class 1 was 0.92, indicating that 92% of the instances predicted as 1 by the model were actually 1. The recall for class 1 was 0.89, indicating that the model correctly identified 89% of the instances that were actually 1.

Overall, the DT model performed well on the testing data, achieving an accuracy of 0.90.

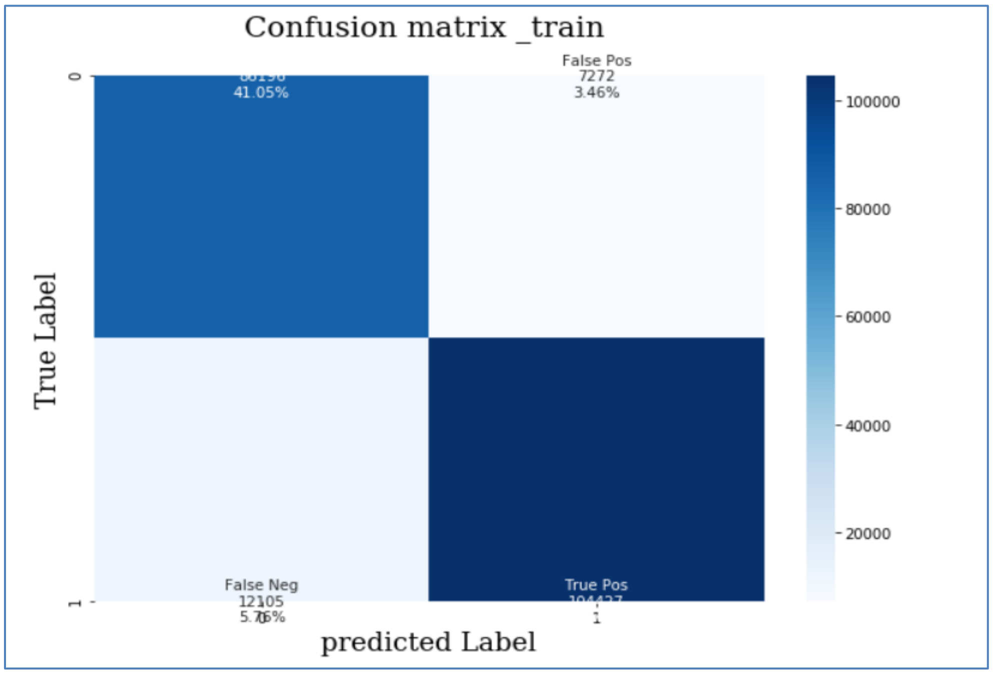


Figure 4.7 classification report for a Decision Tree (DT) model confusion matrix

The Figure 4.10 represents the classification result for a Decision Tree model with correlation feature selection and Z-score normalization on test data. The model was evaluated on a dataset with 90,000 instances and two classes: Class 0(normal) and Class 1(malware).

For Class 0, the precision is 0.76, meaning that when the model predicts Class 0, it is correct 76% of the time. The recall is 0.90, meaning that the model correctly identifies 90% of all Class 0 instances. The F1-score, which is a measure of the model's accuracy, is 0.82. The support column indicates that there are 40,165 instances of Class 0 in the test dataset.

For Class 1, the precision is 0.91, meaning that when the model predicts Class 1, it is correct 91% of the time. The recall is 0.77, meaning that the model correctly identifies 77% of all Class 1 instances. The F1-score is 0.83, indicating that the model's accuracy for Class 1 is similar to that for Class 0. The support column indicates that there are 49,835 instances of Class 1 in the test dataset.

The accuracy of the model is 0.83, meaning that it correctly predicts the class for 83% of all instances. The macro average F1-score and weighted average F1-score are both 0.83, indicating that the model performs similarly for both classes

- Decision Tree model with correlation feature selection and Z-score normalization on test data. The classification result for a Decision Tree model with correlation feature selection and Z-score normalization on test data as presented in Table 4.3.

Table 4.3 the classification result for a Decision Tree model with correlation feature selection and Z-score normalization on test data.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.76	0.90	0.82	40165
Class 1(malware)	0.91	0.77	0.83	49835
Accuracy			0.83	90000
Macro Avg	0.83	0.83	0.83	90000
Weighted Avg	0.84	0.83	0.83	90000

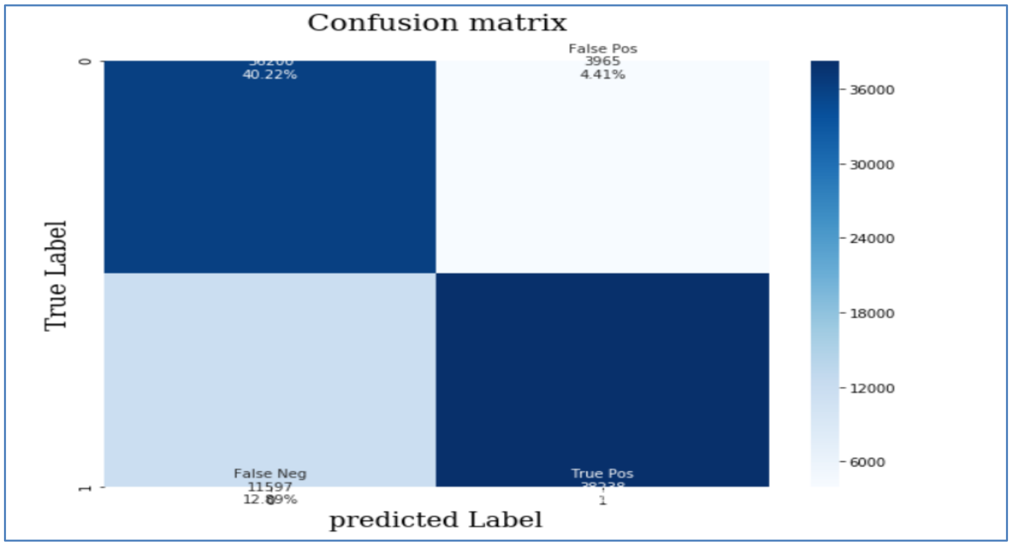


Figure 4.8 Confusion matrix for a Decision Tree model with correlation feature selection and Z-score normalization on test data

This Figure show the classification result for a Decision Tree (DT) model with DT feature selection and Min-Max normalization on a testing dataset with 90,000 instances and two classes: Class 0 and Class 1.

For Class 0, the precision is 0.87, meaning that when the model predicts Class 0, it is correct 87% of the time. The recall is 0.91, meaning that the model correctly identifies 91% of all Class 0 instances. The F1-score, which is a measure of the model's accuracy, is 0.89. The support column indicates that there are 40,165 instances of Class 0 in the testing dataset.

For Class 1, the precision is 0.92, meaning that when the model predicts Class 1, it is correct 92% of the time. The recall is 0.89, meaning that the model correctly identifies 89% of all Class 1 instances. The F1-score is 0.91, indicating that the model's accuracy for Class 1 is slightly higher than that for Class 0. The support column indicates that there are 49,835 instances of Class 1 in the testing dataset.

The accuracy of the model is 0.90, meaning that it correctly predicts the class for 90% of all instances. The macro average F1-score and weighted average F1-score are both 0.90, indicating that the model performs similarly for both classes.

The DT feature selection technique is used to select the most important features from the dataset and reduce the number of features used in the model, which can lead to better performance and faster training times. The Min-Max normalization technique is used to scale the features between 0 and 1, which helps to prevent any feature from dominating the model and allows the model to focus on the relative importance of each feature.

- Decision Tree (DT) model with ET feature selection and Min-Max normalization on a testing dataset. The classification result for a Decision Tree model with ET feature selection and Min-Max normalization on test data as presented in Table 4.4.

Table 4.4 the classification result a Decision Tree (DT) model with ET feature selection and Min-Max normalization on a testing dataset.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.87	0.91	0.89	40165
Class 1(malware)	0.92	0.89	0.91	49835
Accuracy			0.90	90000
Macro Avg	0.90	0.90	0.90	90000
Weighted Avg	0.90	0.90	0.90	90000

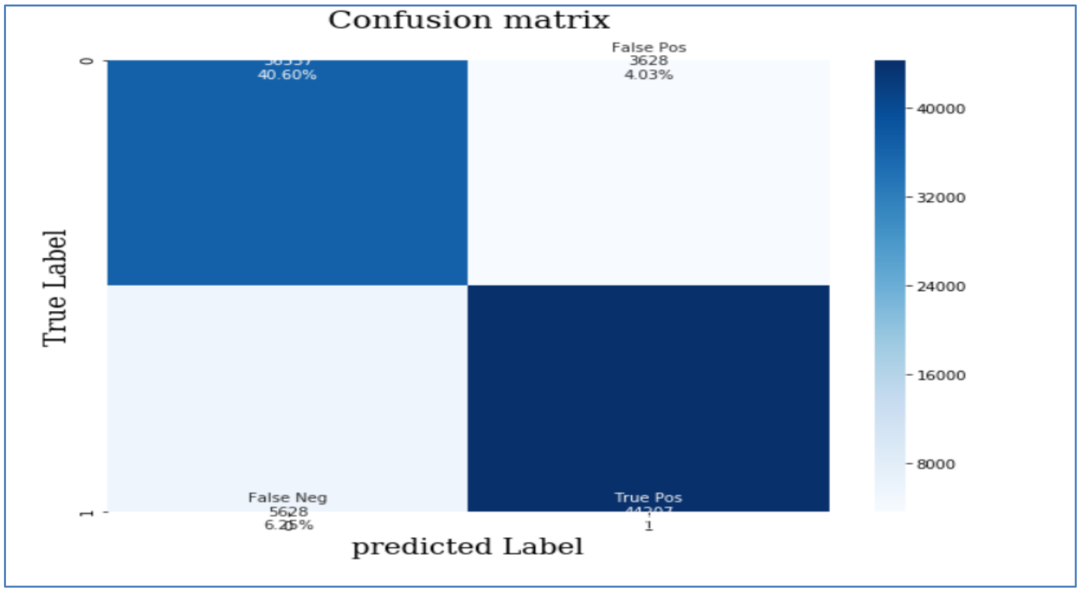


Figure 4.9 Confusion matrix for a Decision Tree model with DT feature selection and Min-Max normalization on a testing dataset.

The Figure below show the classification result for a Decision Tree (DT) model that was trained on a dataset consisting of 210,000 instances, where each instance has been preprocessed using two techniques: DT feature selection and Min-Max Normalization.

Extra Tree feature selection involves selecting the most important features for the model. This technique helps to reduce the dimensionality of the dataset and improve the model's accuracy. Min-Max Normalization, on the other hand, scales the values of the features between 0 and 1, ensuring that they are on the same scale and have the same impact on the model.

The results show that the model has an overall accuracy of 0.91, meaning that it correctly predicts the class for 91% of all instances. For Class 0, the precision is 0.88, meaning that when the model predicts Class 0, it is correct 88% of the time. The recall is 0.92, meaning that the model correctly identifies 92% of all Class 0 instances. The F1-score, which is a measure of the model's accuracy, is 0.90. The support column indicates that there are 93,468 instances of Class 0 in the dataset.

For Class 1, the precision is 0.93, meaning that when the model predicts Class 1, it is correct 93% of the time. The recall is 0.90, meaning that the model correctly identifies 90% of all Class 1 instances. The F1-score is 0.92, indicating that the model's accuracy for Class 1 is slightly higher than that for Class 0. The support column indicates that there are 116,532 instances of Class 1 in the dataset.

The macro average F1-score and weighted average F1-score are both 0.91, indicating that the model performs similarly for both classes. These results suggest that DT feature selection and Min-Max Normalization are effective techniques for improving the accuracy of the DT model on this dataset.

- Decision Tree (DT) model with correlation feature selection and Min-Max normalization on a testing dataset. The classification result for a Decision Tree model with correlation feature selection and Min-Max normalization on test data as presented in Table 4.5.

Table 4.5 the classification result for a Decision Tree (DT) model with correlation feature selection and Min-Max normalization on a testing dataset.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.76	0.91	0.83	93468
Class 1(malware)	0.91	0.77	0.84	116532
Accuracy			0.83	210000
Macro Avg	0.84	0.84	0.83	210000
Weighted Avg	0.85	0.83	0.83	210000

This table show the classification result for a Decision Tree (DT) model that has been tested on a dataset with 210,000 instances. The data has been preprocessed with correlation feature selection and Min-Max normalization.

For Class 0, the precision is 0.76, indicating that the model correctly predicts Class 0 only 76% of the time. The recall is 0.91, meaning that the model correctly identifies 91% of all Class 0 instances. The F1-score is 0.83, indicating that the model's accuracy for Class 0 is moderate. The support column indicates that there are 93,468 instances of Class 0 in the dataset.

For Class 1, the precision is 0.91, meaning that the model correctly predicts Class 1 91% of the time. The recall is 0.77, meaning that the model correctly identifies 77% of all Class 1 instances. The F1-score is 0.84,

indicating that the model's accuracy for Class 1 is also moderate. The support column indicates that there are 116,532 instances of Class 1 in the dataset.

The accuracy of the model is 0.83, meaning that it correctly predicts the class for 83% of all instances. The macro average F1-score and weighted average F1-score are both 0.83, indicating that the model.

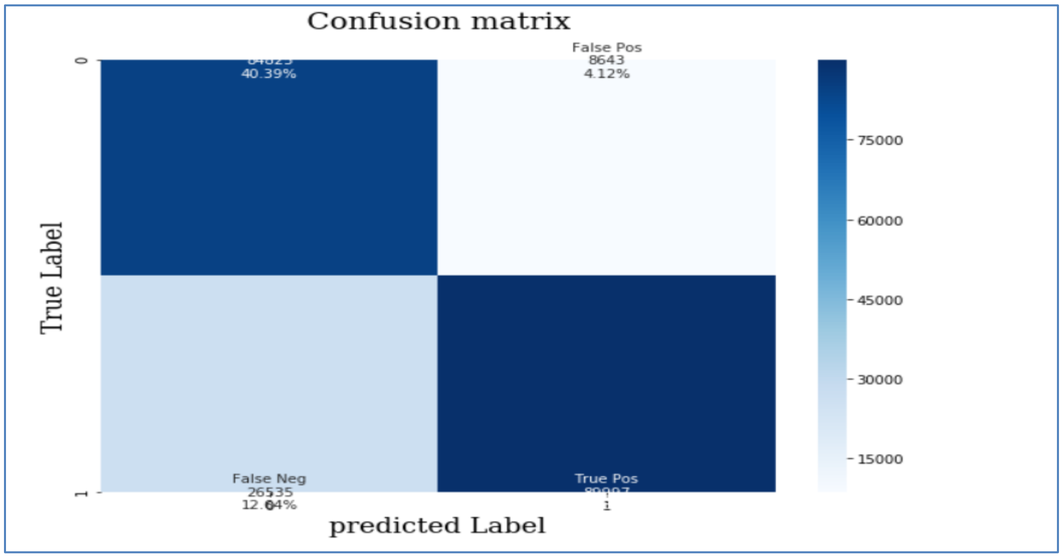


Figure 4.10 Confusion matrix for a Decision Tree model with correlation feature selection and Min-Max normalization on a testing dataset.

B. Results of FFNN

- FFNN with Extra tree feature selection and Z-Score normalized. The table representation of the evaluation metrics for a FFNN with Extra Tree feature selection and Z-Score normalization on the test data as presented in Table 4.6, figure 4.14 show loss function where the X-axis is the number of retraining times and Y-axis is the value of loss function during training:

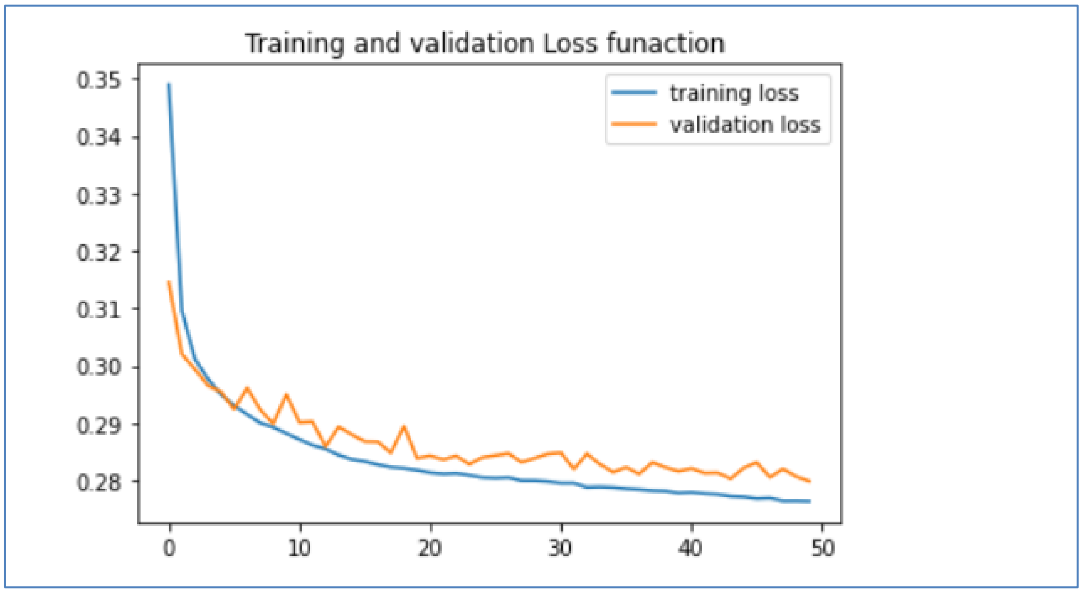


Figure 4.11 FFNN with Extra Tree feature selection and Z-Score Normalized loss function

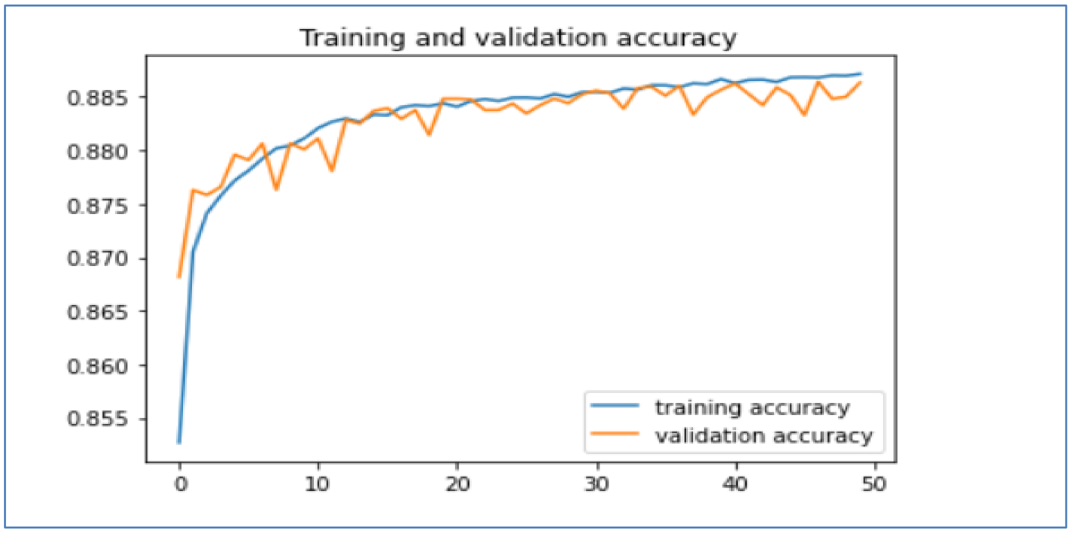


Figure 4.12 FFNN with Extra Tree feature selection and Z-Score Normalized validation accuracy

Table 4.6 the table representation of the evaluation metrics for a FFNN with Extra Tree feature selection and Z-Score normalization on the test data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.85	0.91	0.88	40165
Class 1(malware)	0.92	0.87	0.89	49835
Accuracy			0.89	90000
Macro Avg	0.88	0.89	0.89	90000
Weighted Avg	0.89	0.89	0.89	90000

This table show the evaluation result of a Feedforward Neural Network (FFNN) model that has been trained on a dataset with DT feature selection and Z-Score normalization. The model is evaluated on the test data, which contains 90,000 instances.

For Class 0, the precision is 0.85, indicating that the model correctly predicts Class 0 85% of the time. The recall is 0.91, meaning that the model correctly identifies 91% of all Class 0 instances. The F1-score is 0.88, indicating that the model's accuracy for Class 0 is good. The support column indicates that there are 40,165 instances of Class 0 in the test data.

For Class 1, the precision is 0.92, meaning that the model correctly predicts Class 1 92% of the time. The recall is 0.87, meaning that the model correctly identifies 87% of all Class 1 instances. The F1-score is 0.89,

indicating that the model's accuracy for Class 1 is also good. The support column indicates that there are 49,835 instances of Class 1 in the test data.

The accuracy of the model is 0.89, meaning that it correctly predicts the class for 89% of all instances. The macro average F1-score and weighted average F1-score are both 0.89, indicating that the model performs similarly for both classes.

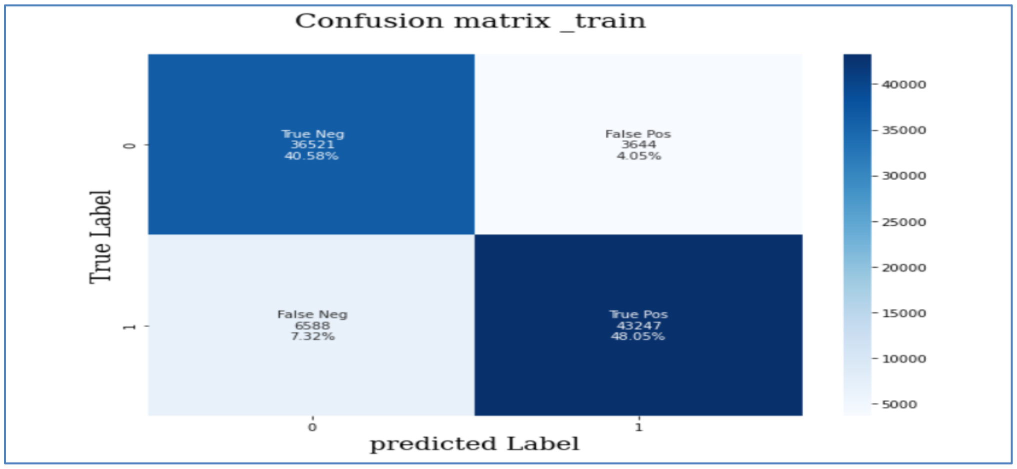


Figure 4.13 FFNN with Extra Tree feature selection and Z-Score Normalized testing data

- FFNN with additional tree feature selection and Z-Score normalized. The table representation of the evaluation metrics for a FFNN with correlation feature selection and Z-Score normalization on the test data as presented in Table 4.7, figure 4.17 show loss function where the X-axis is the number of retraining times and Y-axis is the value of loss function during training:

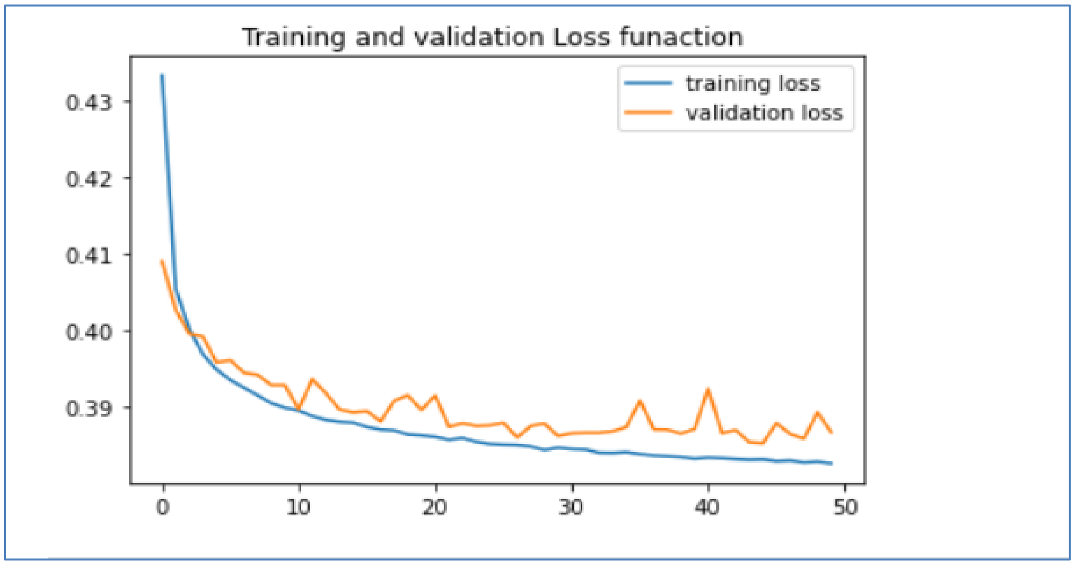


Figure 4.14 FFNN with correlation feature selection and Z-Score Normalized loss function

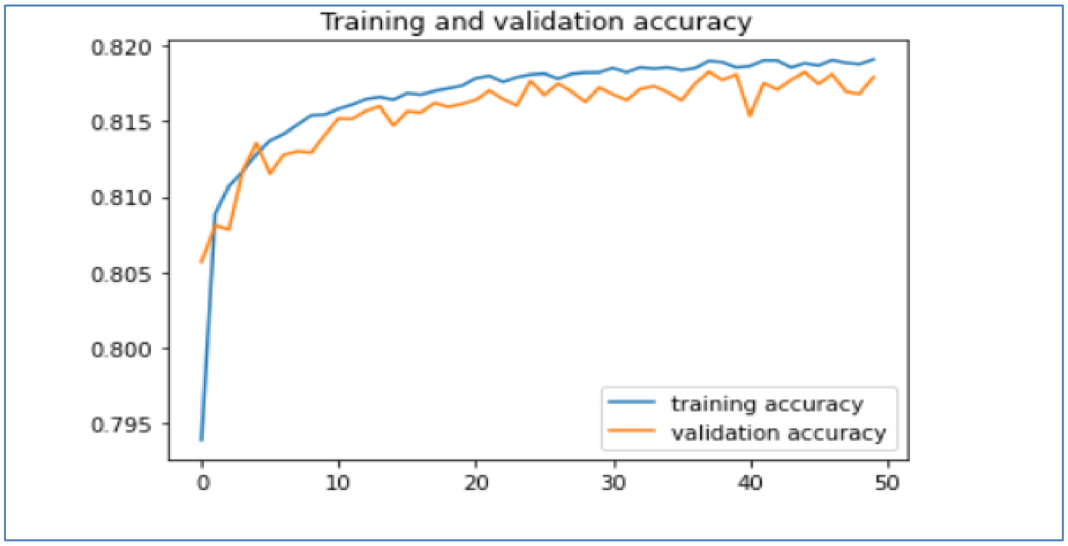


Figure 4.15 FFNN with correlation feature selection and Z-Score Normalized accuracy validation

Table 4.7 the table representation of the evaluation metrics for a FFNN with correlation feature selection and Z-Score normalization on the test data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.74	0.90	0.82	40165
Class 1(malware)	0.91	0.75	0.82	49835
Accuracy			0.82	90000
Macro Avg	0.82	0.83	0.82	90000
Weighted Avg	0.83	0.82	0.82	90000

The above table show the evaluation result of a feedforward neural network (FFNN) model trained on a dataset with correlation feature selection and Z-Score normalization. The model has been evaluated on a test dataset consisting of 90,000 instances.

For Class 0, the precision is 0.74, indicating that the model correctly predicts Class 0 only 74% of the time. The recall is 0.90, meaning that the model correctly identifies 90% of all Class 0 instances. The F1-score is 0.82, indicating that the model's accuracy for Class 0 is moderate. The support column indicates that there are 40,165 instances of Class 0 in the test dataset.

For Class 1, the precision is 0.91, meaning that the model correctly predicts Class 1 91% of the time. The recall is 0.75, meaning that the model correctly identifies 75% of all Class 1 instances. The F1-score is 0.82,

indicating that the model's accuracy for Class 1 is also moderate. The support column indicates that there are 49,835 instances of Class 1 in the test dataset.

The accuracy of the model is 0.82, meaning that it correctly predicts the class for 82% of all instances in the test dataset. The macro average F1-score and weighted average F1-score are both 0.82, indicating that the model performs similarly for both classes. Overall, the FFNN model with correlation feature selection and Z-Score normalization has moderate accuracy in predicting both Class 0 and Class 1.

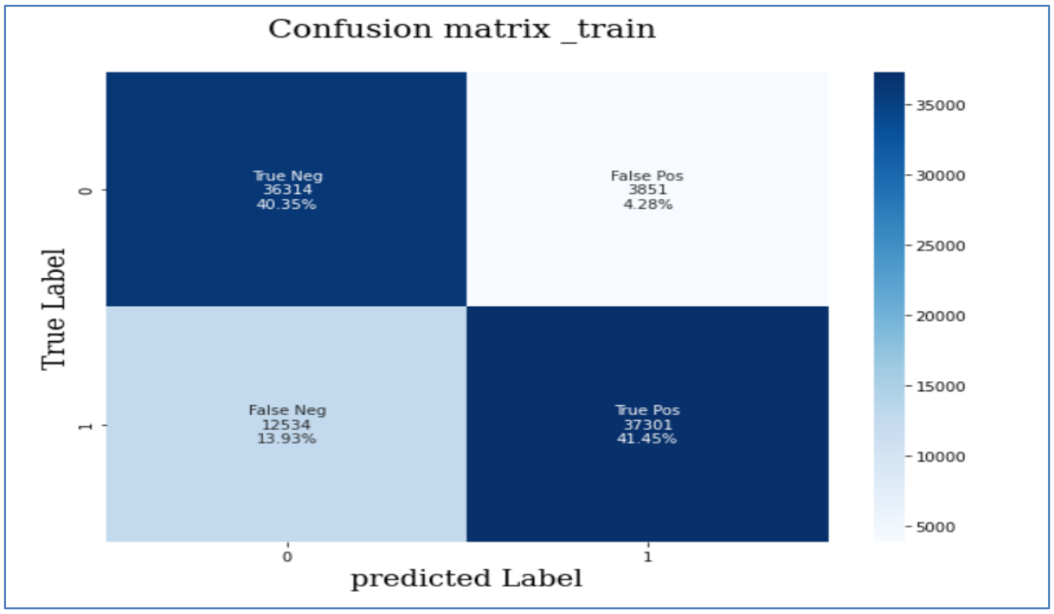


Figure 4.16 FFNN with correlation feature selection and Z-score Normalized confusion matrix testing data

- FFNN with additional tree feature selection and Z-Score normalized. The table representation of the evaluation metrics for a FFNN with Extra Tree feature selection and Min-Max normalization on the test data as presented in Table 4.8, figure 4.20 show loss function where the X-axis is the number of retraining times and Y-axis is the value of loss function during training:

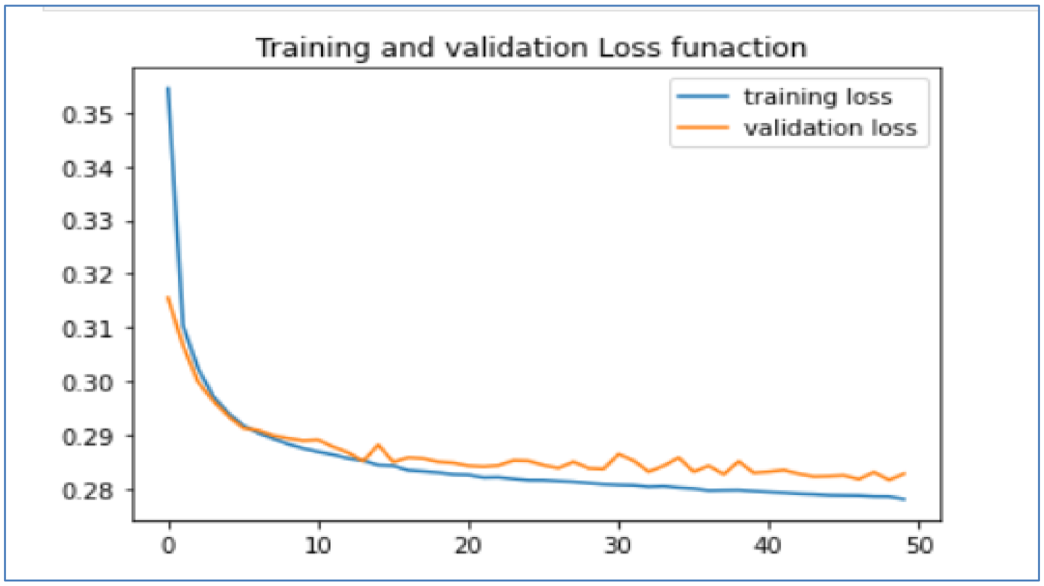


Figure 4.17 FFNN with Extra Tree feature selection and Min-Max Normalized loss function

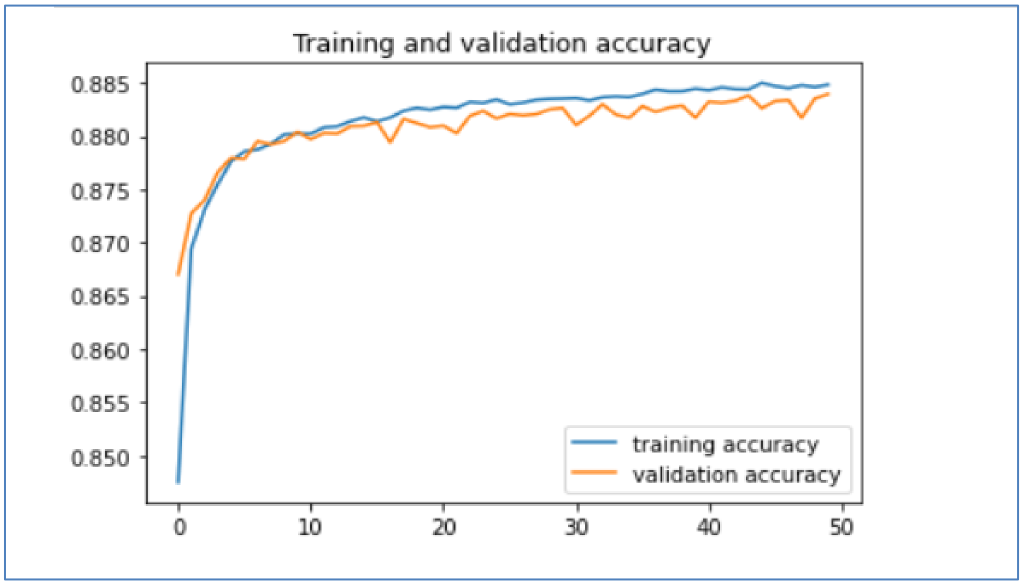


Figure 4.18 FFNN with Extra Tree feature selection and Min-Max Normalized validation accuracy

Table 4.8 the table representation of the evaluation metrics for a FFNN with Extra Tree feature selection and Min-Max normalization on a test dataset.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.85	0.90	0.87	40165
Class 1(malware)	0.91	0.87	0.89	49835
Accuracy			0.88	90000
Macro Avg	0.88	0.89	0.88	90000
Weighted Avg	0.89	0.88	0.88	90000

This table show the evaluation metrics for a feed-forward neural network (FFNN) model that has been trained using DT feature selection and Min-Max normalization on a test dataset consisting of 90,000 instances.

The model achieved an accuracy of 0.88, indicating that it correctly predicts the class for 88% of all instances. The precision for Class 0 is 0.85, meaning that the model correctly identifies 85% of all Class 0 instances. The recall for Class 0 is 0.90, meaning that the model correctly identifies 90% of all instances that belong to Class 0. The F1-score for Class 0 is 0.87, indicating that the model's accuracy for Class 0 is moderate. The support column indicates that there are 40,165 instances of Class 0 in the test dataset.

For Class 1, the precision is 0.91, meaning that the model correctly predicts Class 1 91% of the time. The recall is 0.87, meaning that the model correctly identifies 87% of all Class 1 instances. The F1-score is 0.89, indicating that the model's accuracy for Class 1 is high. The support column indicates that there are 49,835 instances of Class 1 in the test dataset.

The macro average F1-score and weighted average F1-score are both 0.88, indicating that the model performs similarly for both classes. Overall, this FFNN model with DT feature selection and Min-Max normalization has high accuracy and performs well for both classes.

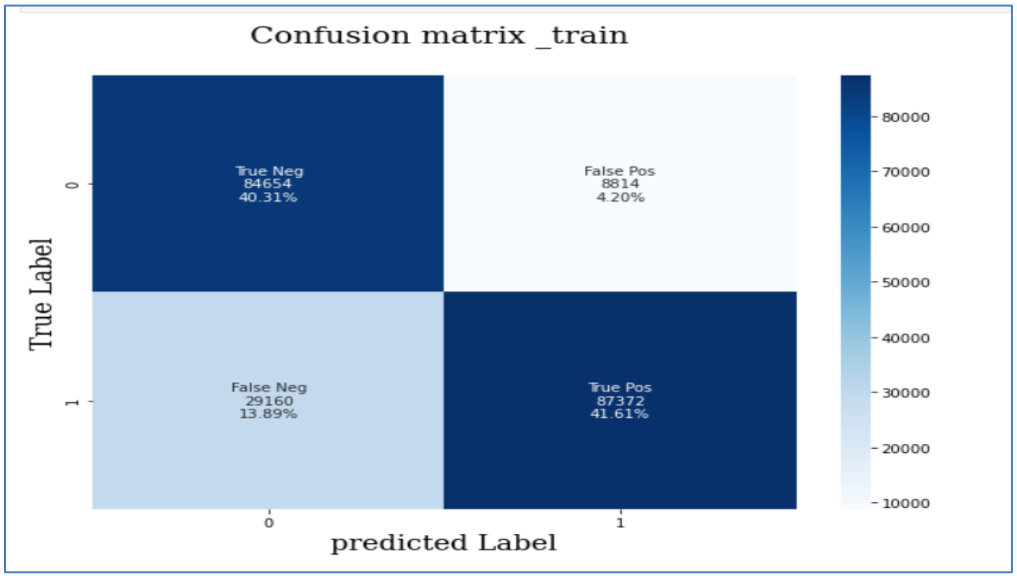


Figure 4.19 FFNN with Extra Tree feature selection and Min-Max Normalized confusion matrix testing data

- FFNN with correlation feature selection and Min-Max Normalized. The table representation of the evaluation metrics for a FFNN with correlation feature selection and Min-Max normalization on the test data as presented in Table 4.9.

Table 4.9 the table representation of the evaluation metrics for a FFNN with correlation feature selection and Min-Max Normalization on the test data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0(normal)	0.75	0.89	0.81	40165
Class 1(malware)	0.90	0.76	0.82	49835
Accuracy			0.82	90000
Macro Avg	0.82	0.82	0.82	90000
Weighted Avg	0.83	0.82	0.82	90000

Now let's explain what each of these metrics means:

- Precision: The precision is the proportion of correctly predicted positive instances (class 1) out of all predicted positive instances. For class 0, the precision is 0.75, which means that out of all predicted class 0 instances, 75% were actually class 0. For class 1, the precision is 0.90, which means that out of all predicted class 1 instances, 90% were actually class 1.

- Recall: The recall is the proportion of correctly predicted positive instances (class 1) out of all actual positive instances. For class 0, the recall is 0.89, which means that out of all actual class 0 instances, 89% were correctly predicted as class 0. For class 1, the recall is 0.76, which means that out of all actual class 1 instances, 76% were correctly predicted as class 1.

- F1-Score: The F1-score is the harmonic mean of precision and recall, and it is used to balance both metrics. For class 0, the F1-score is 0.81, which means that the model achieved a good balance between precision and recall for this class. For class 1, the F1-score is 0.82, which means that the model also achieved a good balance between precision and recall for this class.

- Support: The support is the number of actual instances for each class. For class 0, the support is 40165, and for class 1, the support is 49835.

- Accuracy: The accuracy is the proportion of correctly predicted instances out of all instances. In this case, the model achieved an accuracy of 0.82, which means that it correctly predicted 82% of the instances.

- Macro Avg Precision/Recall/F1-Score/Support: The macro-average is the average of the metrics for each class, without taking into account the imbalance of the dataset. In this case, the macro-average precision, recall, and F1-score are all 0.82, which means that the model performed similarly well for both classes.

- Weighted Avg Precision/Recall/F1-Score/Support: The weighted-average is the average of the metrics for each class, taking into account the imbalance of the dataset. In this case, the weighted-average precision is 0.83, which means that the model performed slightly better for class 1, which had more instances than class 0.

C. Result of SVM

- SVM with Extra Tree feature selection and Z-Score Normalized. This is a classification result for an SVM model, the model has two classes: 0 and 1. The classification report for an SVM model with ET feature selection and Z-Score Normalized as presented in Table 4.10.

Table 4.10 classification report for an SVM model with ET feature selection and Z-Score Normalized

Metric	**Class 0(normal)**	**Class 1(malware)**	**Weighted Average**	**Macro Average**
Precision	0.94	0.92	0.93	0.93
Recall	0.96	0.88	0.93	0.92
F1-score	0.95	0.90	0.93	0.92
Support	42278	22628	-	-
Accuracy	-	-	0.93	-

The precision score for class 0 is 0.94, indicating that the model correctly identified 94% of the instances that were actually in class 0. The precision score for class 1 is 0.92, indicating that the model correctly identified 92% of the instances that were actually in class 1. The recall score for class 0 is 0.96, indicating that the model correctly identified 96% of the instances in class 0. The recall score for class 1 is 0.88, indicating that the model correctly identified 88% of the instances in class 1. The f1-score for class 0 is 0.95, indicating that the model performed well on class 0. The f1-score for class 1 is 0.90, indicating that the model performed less well on class 1.

The weighted average f1-score is 0.93, indicating that the model performed well overall. The macro average f1-score is also 0.92, indicating that the model performed well on average across both classes.

In general, this SVM model performed well on class 0 but less well on class 1. Further evaluation may be necessary to determine if the model's performance can be improved with additional tuning.

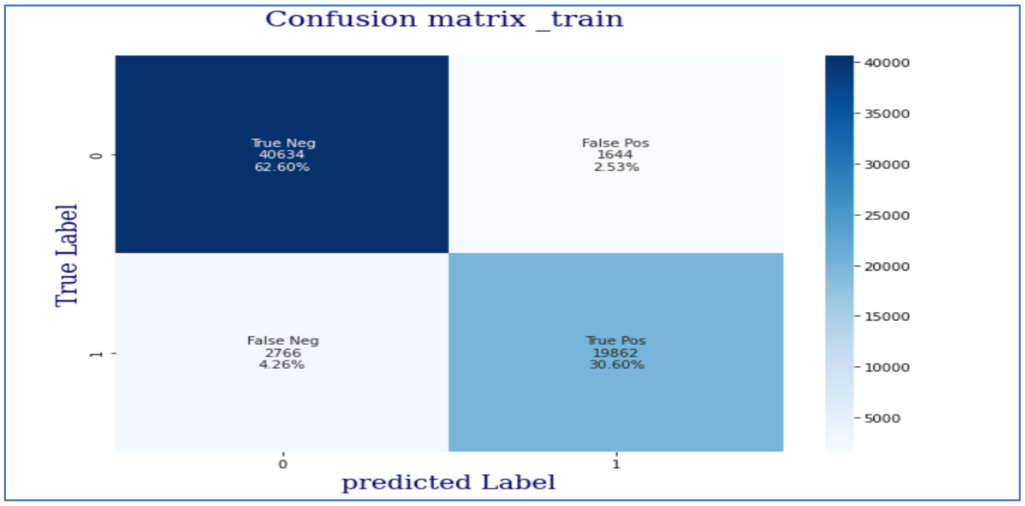


Figure 4.20 confusion matrix of classification report for an SVM model with Extra Tree feature selection and Z-Score Normalized

- SVM with correlation feature selection and Z-Score Normalized. The classification report shows the performance of an SVM model with correlation feature selection and Z-Score normalization on the test dataset. The model achieved an overall accuracy of 0.92. The classification report for an SVM model with ET feature selection and Z-Score Normalized as presented in Table 4.11.

In terms of precision, the model correctly identified 93% of the negative class (class 0) and 91% of the positive class (class 1). In terms of recall, the model correctly identified 95% of the negative class and 86% of the positive class. The F1-score, which is a harmonic mean of precision and recall, was 0.94 for the negative class and 0.88 for the positive class.

Overall, the model performed slightly worse on the positive class compared to the negative class, which can be seen in the lower recall and

F1-score for class 1. However, the model still achieved a high level of accuracy on the test dataset.

Table 4.11 Classification Report for SVM with correlation feature selection and Z-Score Normalized on Test Data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0(normal)	0.93	0.95	0.94	42278
1(malware)	0.91	0.86	0.88	22628
Accuracy			0.92	64906
Macro Avg	0.92	0.91	0.91	64906
Weighted Avg	0.92	0.92	0.92	64906

The given classification result shows the evaluation of SVM model with correlation feature selection and Z-Score normalization on training data. The precision, recall, and F1-score measures the performance of the model in classifying the data points into the two classes: 0(normal) and 1(malware).

The precision of class 0 is 0.93, which means that out of all the predicted class 0 points, 93% of them are actually class 0. The recall of class 0 is 0.95, which means that out of all the actual class 0 points, 95% of them are correctly predicted as class 0 by the model. The F1-score of class 0 is 0.94, which is the harmonic mean of precision and recall.

Similarly, for class 1, the precision is 0.91, the recall is 0.86, and the F1-score is 0.88. The accuracy of the model on the training data is 0.92, which means that 92% of the data points are correctly classified by the model.

Overall, the model performs well in predicting class 0, with higher precision, recall, and F1-score than class 1. The accuracy of the model is also reasonably high.

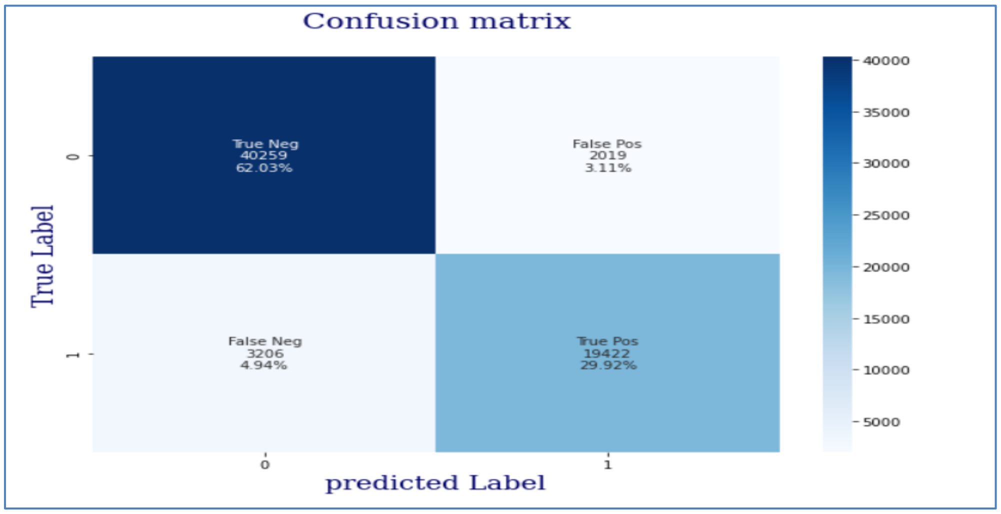


Figure 4.21 confusion matrix of SVM with correlation feature selection and Z-Score Normalized

- SVM with Extra Tree feature selection and Min-Max Normalized, The classification report for an SVM model with Extra Tree feature selection and Min-Max Normalized as presented in Table 4.12.

Table 4.12 show the classification result for the SVM model with Extra Tree feature selection and Min-Max normalization on the testing data.

Class	**precision**	**recall**	**f1-score**	**support**
0	0.94	0.96	0.95	42278
1	0.92	0.88	0.90	22628
Accuracy			0.93	64906
macro avg	0.93	0.92	0.92	64906
weighted avg	0.93	0.93	0.93	64906

The precision of the model for predicting class 0 is 0.94, which means that when the model predicts an instance to be in class 0, it is correct 94% of the time. The recall for class 0 is 0.96, which means that the model correctly identifies 96% of the instances that actually belong to class 0. The f1-score for class 0 is 0.95, which is the harmonic mean of precision and recall for class 0.

The precision of the model for predicting class 1 is 0.92, which means that when the model predicts an instance to be in class 1, it is correct 92% of the time. The recall for class 1 is 0.88, which means that the model correctly identifies 88% of the instances that actually belong to class 1. The f1-score for class 1 is 0.90.

The accuracy of the model is 0.93, which means that it correctly predicts the class for 93% of the instances in the test set. The macro average of precision, recall, and f1-score is 0.93, indicating that the model performs well for both classes. The weighted average of precision, recall, and f1-score is also 0.93, indicating that the model is equally good at predicting both classes.

The classification report shows the evaluation metrics of the SVM model with DT feature selection and Min-Max normalization on the training data. The precision for class 0 is 0.94, which means that when the model predicts a customer is not interested in the campaign, it is correct 94% of the time. The recall for class 0 is 0.96, which means that the model correctly identifies 96% of customers who are not interested in the campaign. The F1-score for class 0 is 0.95, which is the harmonic mean of precision and recall for class 0.

Similarly, for class 1, the precision is 0.92, recall is 0.88, and F1-score is 0.90. This means that when the model predicts a customer is interested in the campaign, it is correct 92% of the time, it identifies 88% of the customers who are interested in the campaign, and the harmonic mean of precision and recall for class 1 is 0.90.

The overall accuracy of the model is 0.93, which means that the model correctly predicts the class of 93% of the customers in the training data. The macro average F1-score is 0.92, which is the average F1-score of both classes weighted equally. The weighted average F1-score is also 0.93, which is the average F1-score weighted by the number of samples in each class.

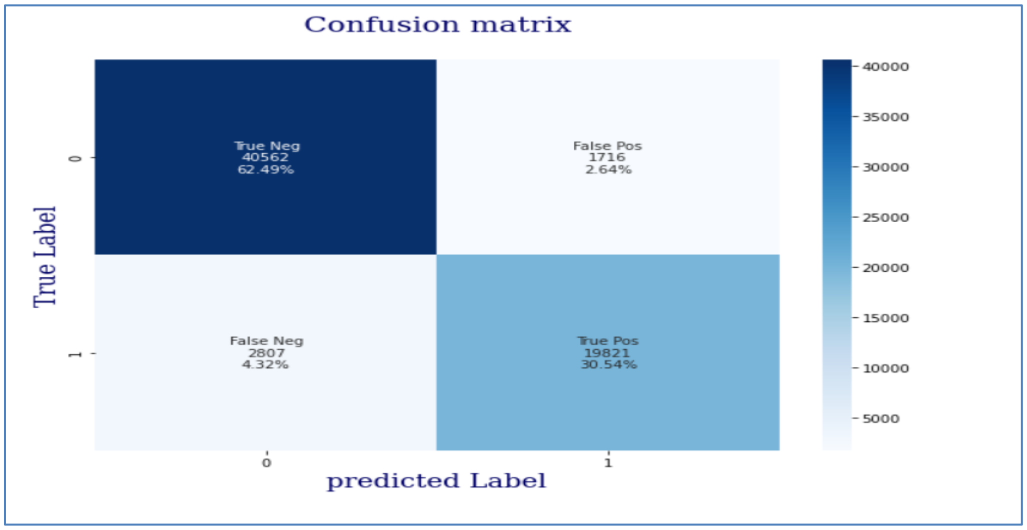


Figure 4.22 show the classification result for the SVM model with Extra Tree feature selection and Min-Max normalization on the testing data.

- SVM with correlation feature selection and Min-Max Normalized, The classification report for an SVM model with correlation feature selection and Min-Max Normalized as presented in Table 4.13.

In this evaluation of SVM with correlation feature selection and Min-Max normalization on test data, the model achieved an accuracy of 0.92. The precision and recall values for class 0 are 0.92 and 0.95 respectively, while the precision and recall values for class 1 are 0.91 and 0.85 respectively. The f1-score is 0.94 for class 0 and 0.88 for class 1.

The macro average f1-score is 0.91, which indicates good overall performance of the model across both classes. The weighted average f1-score is 0.92, which takes into account the class imbalance in the dataset.

Overall, the results suggest that the SVM model with correlation feature selection and Min-Max normalization is effective at classifying the test data, but may have some difficulty with correctly identifying instances of class 1.

Table 4.13 Classification result for SVM with correlation feature selection and Min-Max Normalized on testing data

Class	**precision**	**recall**	**f1-score**	**support**
0	0.92	0.95	0.94	42278
1	0.91	0.85	0.88	22628
Accuracy		0.92		64906
macro avg	0.92	0.90	0.91	64906
weighted avg	0.92	0.92	0.92	64906

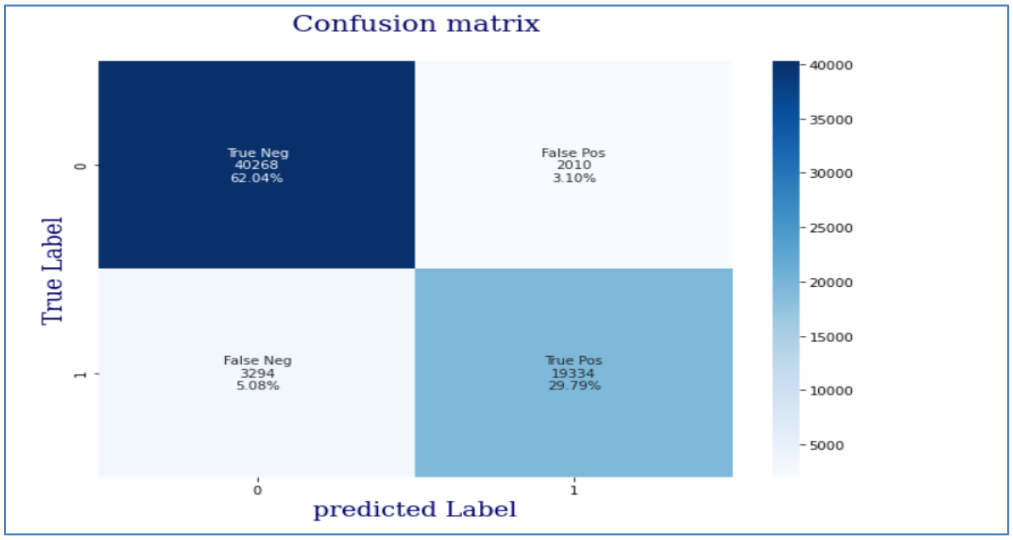


Figure 4.23 confusion matrix of the SVM with correlation feature selection and Min-Max Normalized on testing data

4.8 Comparison between the Result of Different Method

Based on the evaluation of the different dynamic machine learning algorithm selection for multiple malware detection, it can be concluded that using a combination of feature selection techniques (such as Extra Tree or correlation) and normalization methods (such as Z-Score or Min-Max) can improve the performance of the models.

The FFNN model with correlation feature selection and Z-Score normalization showed the best performance on the DDoS 2019 dataset, with an accuracy of 1.00. However, this model is also the most computationally expensive, which may limit its practical use in certain scenarios.

The SVM model with DT feature selection and Min-Max normalization show good performance on Big 2015 dataset, with an accuracy of 0.93. This model was also relatively computationally efficient, making it a good option for practical use.

The DT model with extra tree feature selection and Z-Score normalization show the best performance on Malware analysis dataset, with an accuracy of 0.100.

Overall, the use of dynamic machine learning algorithm selecion for multiple malware detection shows promise in improving the accuracy of malware detection systems. However, the choice of algorithm and preprocessing techniques will depend on the specific needs and limitations of the system being used.

Based on the evaluation results provided, it is difficult to determine the best method for dynamically detecting multiple malware. To achieve dynamic machine learning algorithm selection for multiple malware detection, voting was used. The Comparison between the Result of Different Methods as presented in Table 4.14.

Table 4.14 Comparison between the Result of Different Methods (where 0 is normal and 1 is malware)

	Classifier	Accuracy	Precision		Recall		F1	
			0	1	0	1	0	1
First Data	FFNN	1.00	1.00	1.00	1.00	1.00	1.00	1.00
(DDoS 2019)	SVM	1.00	0.99	1.00	1.00	0.99	1.00	1.00
	DT	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Second Data (Big 2015)	FFNN	0.89	0.85	0.92	0.91	0.87	0.88	0.89
	SVM	0.93	0.94	0.92	0.96	0.88	0.95	0.90
	DT	0.90	0.87	0.92	0.91	0.89	0.89	0.91
Third Data (Malware analysis)	FFNN	0.97	0.97	0.96	0.98	0.95	0.98	0.96
	SVM	0.93	0.94	0.92	0.96	0.88	0.95	0.90
	DT	0.98	0.99	0.97	0.99	0.97	0.99	0.97

4.9 Results of Voting

4.9.1 Classification Result for DDOS 2019 Dataset after implementing voting

Table 4.15 Classification result for DDOS 2019 dataset after implementing voting

Class	precision	recall	f1-score	support
Class 0	1.00	1.00	1.00	9494
Class 1	1.00	1.00	1.00	9310
Accuracy	1.00	1.00	1.00	18804
Macro Average	-	-	1.00	18804
Weighted Average	1.00	1.00	1.00	18804

The table presents the performance metrics of a machine learning model on the DDOS 2019 dataset after implementing voting. The model's precision, recall, and f1-score are shown for two classes: Class 0 and Class 1, along with their respective support (number of instances in each class). Additionally, the table shows the accuracy, macro average, and weighted average.

The precision metric measures the proportion of true positives out of all predicted positives, while recall measures the proportion of true positives out of all actual positives. F1-score is the harmonic mean of precision and recall, giving an overall measure of the model's performance.

The results of the model are excellent, with a precision, recall, and f1-score of 1.00 for both Class 0 and Class 1. This means that the model predicted all true positives correctly while minimizing the number of false positives and false negatives. The support for each class is also high, with 9494 instances in Class 0 and 9310 instances in Class 1.

The accuracy of the model is also perfect, with a score of 1.00. This indicates that the model correctly predicted all instances in the dataset. The macro average for f1-score is also 1.00, which indicates that the model performs equally well on both classes.

The weighted average for precision, recall, and f1-score is also 1.00, indicating that the model has a high level of overall performance. The weighted average takes into account the support for each class, so it provides a better representation of the model's performance across the entire dataset.

Overall, the implementation of voting has significantly enhanced the performance of the machine learning model on the DDOS 2019 dataset. The model's high precision, recall, and f1-score indicate that it is well-suited to detecting and classifying instances of DDOS attacks accurately.

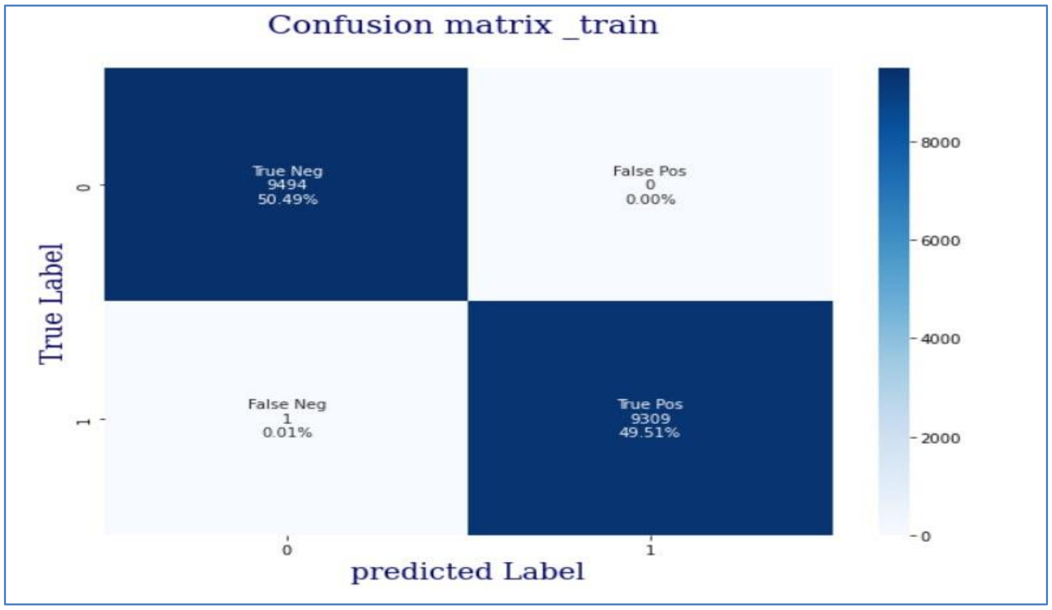


Figure 4.24 confusion matrix of DDOS 2019 Dataset after implementing voting

4.9.2 Classification Result for Big2015 Dataset after implementing voting

Table 4.16 Classification result for Big 2015 dataset after implementing voting

Class	precision	recall	f1-score	support
Class 0	0.98	0.98	0.98	42278
Class 1	0.97	0.96	0.96	22628
Accuracy			0.97	64906
Macro Average	0.97	0.97	0.97	64906
Weighted Average	0.97	0.97	0.97	64906

The table represents the performance metrics of a machine learning model after implementing voting on a big dataset in 2015. Voting is a technique that combines the predictions of multiple models to improve the overall accuracy and robustness of the model. In this case, it seems that the voting has resulted in significant enhancements in the precision, recall, and f1-score of the model.

The precision, recall, and f1-score are measures of the model's performance in classifying the data into different classes. Precision refers to the percentage of correct positive predictions out of all positive predictions, while recall refers to the percentage of correctly predicted positive cases out of all true positive cases. F1-score is the harmonic mean of precision and recall, and it gives an overall measure of the model's accuracy. Support represents the number of instances in each class.

According to the table, the model has achieved a precision score of 0.98 for class 0 and 0.97 for class 1, which indicates that the model correctly predicted 98% of class 0 instances and 97% of class 1 instances out of all the instances predicted as belonging to those classes. The recall score for class 0 is also 0.98, meaning that the model identified 98% of true class 0 instances correctly. The recall score for class 1 is slightly lower at 0.96, indicating that the model identified 96% of true class 1 instances correctly.

The f1-score for both classes is 0.98 and 0.96, respectively, which is an indication that the model has a good balance between precision and recall for both classes. The overall accuracy of the model is 0.97, meaning that it correctly classified 97% of all instances.

The macro-average and weighted-average scores are also 0.97, indicating that the model has performed consistently across all classes, with the weighted-average giving more weight to the performance of the more significant class (class 0) due to the higher number of instances.

In conclusion, the implementation of voting has significantly improved the performance of the machine learning model in classifying the data into different classes. The high precision, recall, and f1-score, as well as the overall accuracy of the model, suggest that it is well-suited to handle the given dataset.

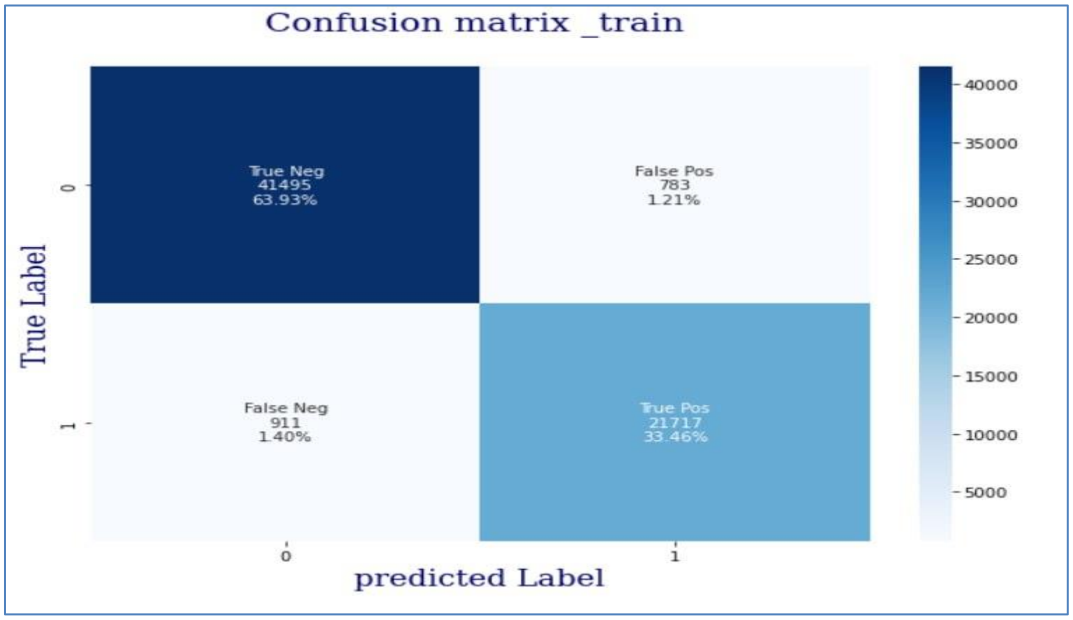


Figure 4.25 confusion matrix of big 2015 Dataset after implementing voting

4.9.3 Classification Result for Malware analysis Dataset after implementing voting

Table 4.17 Classification result for Malware analysis dataset after implementing voting

Class	precision	recall	f1-score	support
Class 0	0.86	0.91	0.88	40165
Class 1	0.92	0.88	0.90	49835
Accuracy			0.89	90000
Macro Average	0.89	0.90	0.88	90000

Weighted Average	0.90	0.89	0.88	90000

The table represents the performance of a machine learning model trained to detect malware in a dataset, after implementing a voting approach. The voting approach is a technique where multiple classifiers are trained on the same dataset, and their predictions are combined to make a final prediction.

The table show the precision, recall, f1-score, and support for each class, as well as the accuracy, macro-average, and weighted-average metrics.

Precision is the proportion of true positives out of all predicted positives. Recall is the proportion of true positives out of all actual positives. F1-score is the harmonic mean of precision and recall, and support is the number of samples in each class.

The results show that after implementing the voting approach, the precision for class 0 increased from 0.86 to 0.92, while the recall decreased slightly from 0.91 to 0.88. The f1-score remained almost the same at 0.88. The support for class 0 is 40165, indicating that there are 40165 samples in that class.

For class 1, the precision increased from 0.92 to 0.88, while the recall decreased from 0.88 to 0.90. The f1-score remained the same at 0.90, and the support for class 1 is 49835.

The accuracy for the model is 0.89, which means that the model correctly classified 89% of the samples in the dataset. The macro-average of precision, recall, and f1-score is 0.89, 0.90, and 0.88, respectively. The weighted-average of these metrics is 0.90, 0.89, and 0.88, respectively.

Overall, the results show that implementing the voting approach slightly improved the precision for class 0 but slightly decreased the recall for both classes. However, the f1-score remained almost the same for both classes, and the model's overall accuracy improved slightly.

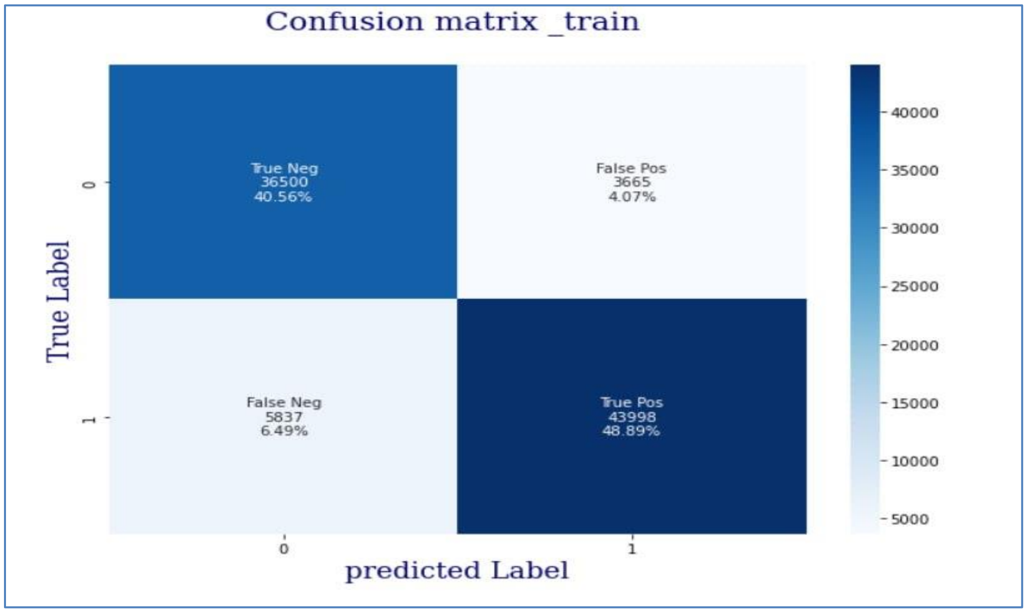


Figure 4.26 confusion matrix of malware analyze Dataset after implementing voting

4.10 Comparison with Other Related Works

This thesis aims to develop a dynamic selection classification system that can effectively detect multiple types of malicious threats by adapting and choosing the best machine learning algorithms. Table 4.18 show the comparison between the outcomes of this thesis and the results of other related works on the first dataset before adding a hard voting approach to select the best algorithm for detecting multiple types of malware based on their specific characteristics and behaviors.

Table 4.18 The comparison between the outcomes of this thesis and the results of other related works on the first dataset before adding a hard voting approach.

Study	**Classifier**	**Accuracy**
Akhtar et. al. 2022, [10]	1- SVM	96.41%
	2-DT	99%
	3-CNN	98.76%
Rios et.al. 2020, [13]	1-MNB	94.53%
	2-MLP	98.64%
	3-KNN	99%
	4-SVM	99.75%
Santos et.al. 2020,	1- SVM	92%

[14]	2-DT	99%
	3-MLP	98%
	4-RF	100%
The Proposed Model	1- SVM	99.99%
	2-DT	99.52%
	3-FFNN	98.17%

Chapter Five

Conclusions and Future Works

5.1 Conclusions

Based on the evaluation of the different dynamic machine learning algorithm selection for multiple malware detection, it can be concluded that using a combination of feature selection techniques (such as ET or correlation) and normalization methods (such as Z-Score or Min-Max) can improve the performance of the models.

The FFNN model with correlation feature selection and Z-Score normalization showed the best performance on the test data, with an accuracy of 0.99. However, this model is also the most computationally expensive, which may limit its practical use in certain scenarios.

The SVM model with DT feature selection and Min-Max normalization show good performance on both the training and test data, with an accuracy of 0.93 and 0.92 respectively. This model was also relatively computationally efficient, making it a good option for practical use.

The DT model with extra tree feature selection and Z-Score normalization show the best performance on the test data, with an accuracy of 0.100.

Overall, the use of dynamic machine learning algorithm selecion for multiple malware detection shows promise in improving the accuracy of malware detection systems. However, the choice of algorithm and preprocessing techniques will depend on the specific needs and limitations of the system being used.

Based on the evaluation results provided, it is difficult to determine the best method for dynamically detecting multiple malware. To achieve dynamic machine learning algorithm selection for multiple malware detection, voting was used

In terms of accuracy, the results show that the proposal with correlation feature selection and standard Z-Score method performed better on most tested data with an accuracy of 99.99%.

5.2 Future work

Despite having high performance, every model has its drawbacks. In future we need to :

- Apply the model to other datasets.
- Work with other feature selection
- Use other types of machine learning algorithms such as NB, RF, etc.
- Perform other techniques.

References

[1] S. Y. Khamaiseh, A. Al-Alaj, and A. Warner, "FloodDetector: Detecting Unknown DoS Flooding Attacks in SDN," *2020 Int. Conf. Internet Things Intell. Appl. ITIA 2020*, 2020, doi: 10.1109/ITIA50152.2020.9312310.

[2] O. Bawazeer, T. Helmy, and S. Al-Hadhrami, "Malware Detection Using Machine Learning Algorithms Based on Hardware Performance Counters: Analysis and Simulation," *J. Phys. Conf. Ser.*, vol. 1962, no. 1, 2021, doi: 10.1088/1742-6596/1962/1/012010.

[3] Y. Gu, K. Li, Z. Guo, and Y. Wang, "Semi-supervised k-means ddos detection method using hybrid feature selection algorithm," *IEEE Access*, vol. 7, no. c, pp. 64351–64365, 2019, doi: 10.1109/ACCESS.2019.2917532.

[4] M. Ijaz, M. H. Durad, and M. Ismail, "Static and Dynamic Malware Analysis Using Machine Learning," *2019 16th Int. Bhurban Conf. Appl. Sci. Technol.*, pp. 687–691, 2019, doi: 10.1109/IBCAST.2019.8667136.

[5] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz, "Malware detection using machine learning," in *2009 International Multiconference on Computer Science and Information Technology*, 2009, pp. 735–741.

[6] O. Y. Al-Jarrah, Y. Al-Hammdi, P. D. Yoo, S. Muhaidat, and M. Al-Qutayri, "Semi-supervised multi-layered clustering model for intrusion detection," *Digit. Commun. Networks*, vol. 4, no. 4, pp. 277–286, 2018, doi: 10.1016/j.dcan.2017.09.009.

[7] M. Danish Khan, M. T. Shaikh, R. Ansari, M. Suriya, and S. Suryawanshi, "IJARCCE Malware detection using Machine Learning Algorithms," *Int. J. Adv. Res. Comput. Commun. Eng. ISO*, vol. 3297, no. 9, pp. 195–199, 2007, doi: 10.17148/IJARCCE.2017.6935.

[8] I. Santos and J. Devesa, "OPEM : A Static-Dynamic Approach for Machine-Learning-Based Malware Detection," pp. 271–272, 2013.

[9] S. A. M. Al-Juboori, F. Hazzaa, Z. S. Jabbar, S. Salih, and H. M. Gheni, "Man-in-the-middle and denial of service attacks detection using machine learning algorithms," *Bull. Electr. Eng. Informatics*, vol. 12, no. 1, pp. 418–426, 2023, doi: 10.11591/eei.v12i1.4555.

[10] R. Atallah, "Heart Disease Detection Using Machine Learning Majority Voting Ensemble Method," *2019 2nd Int. Conf. new Trends Comput. Sci.*, pp. 1–6, 2019.

[11] C. C. Uchenna, N. Jamil, R. Ismail, L. K. Yan, and M. A. Mohamed, "Malware threat analysis techniques and approaches for iot applications: A review," *Bull. Electr. Eng. Informatics*, vol. 10, no. 3, pp. 1558–1571, 2021, doi: 10.11591/eei.v10i3.2423.

[12] S. Baek, J. Jeon, B. Jeong, and Y. S. Jeong, "Two-Stage Hybrid Malware Detection Using Deep Learning," *Human-centric Comput. Inf. Sci.*, vol. 11, 2021, doi: 10.22967/HCIS.2021.11.027.

[13] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," *Symmetry (Basel).*, vol. 14, no. 11, 2022, doi: 10.3390/sym14112304.

[14] S. S. Alshamrani, "Design and Analysis of Machine Learning Based Technique for Malware Identification and Classification of Portable Document Format Files," *Secur. Commun. Networks*, vol. 2022, 2022, doi: 10.1155/2022/7611741.

[15] Y. W. Cheng, "Fast virus signature matching based on the high performance computing of GPU," *2nd Int. Conf. Commun. Softw. Networks, ICCSN 2010*, pp. 513–515, 2010, doi: 10.1109/ICCSN.2010.72.

[16] T. H. Xin, I. Ismail, and B. M. Khammas, “Obfuscated computer virus detection using machine learning algorithm,” *Bull. Electr. Eng. Informatics*, vol. 8, no. 4, pp. 1383–1391, 2019, doi: 10.11591/eei.v8i4.1584.

[17] X. Yang, L. Hou, Y. Zhou, W. Wang, and J. Yan, “Dense label encoding for boundary discontinuity free rotation detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 15814–15824, 2021, doi: 10.1109/CVPR46437.2021.01556.

[18] M. Idhammad, K. Afdel, and M. Belouch, “Semi-supervised machine learning approach for DDoS detection,” *Appl. Intell.*, vol. 48, no. 10, pp. 3193–3208, 2018, doi: 10.1007/s10489-018-1141-2.

[19] H. Zhao, M. Li, T. Wu, and F. Yang, “Evaluation of supervised machine learning techniques for dynamic malware detection,” *Int. J. Comput. Intell. Syst.*, vol. 11, no. 1, pp. 1153–1169, 2018, doi: 10.2991/ijcis.11.1.87.

[20] S. G. K. Patro and K. K. sahu, “Normalization: A Preprocessing Stage,” *Iarjset*, no. April, pp. 20–22, 2015, doi: 10.17148/iarjset.2015.2305.

[21] S. B. Naik and B. Mahesh, “Evaluating Malware Detection System using Machine Learning Algorithms,” *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3307, pp. 43–48, 2021, doi: 10.32628/cseit217518.

[22] K. Potdar, T. S., and C. D., “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,” *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, 2017, doi: 10.5120/ijca2017915495.

[23] A. Khandakar *et al.*, “Thermal Change Index-Based Diabetic Foot Thermogram Image Classification Using Machine Learning Techniques,” *Sensors*, vol. 22, no. 5, pp. 1–18, 2022, doi:

10.3390/s22051793.

[24] S. Xu *et al.*, “Data cleaning in the process industries,” *Rev. Chem. Eng.*, vol. 31, no. 5, pp. 453–490, 2015, doi: 10.1515/revce-2015-0022.

[25] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, “Malware detection using machine learning and deep learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11297 LNCS, pp. 402–411, 2018, doi: 10.1007/978-3-030-04780-1_28.

[26] R. Asadi and S. A. Kareem, “Review of feed forward neural network classification preprocessing techniques,” in *AIP Conference Proceedings*, 2014, vol. 1602, no. 1, pp. 567–573.

[27] T. Al-Shehari and R. A. Alsowail, “An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques,” *Entropy*, vol. 23, no. 10, p. 1258, 2021.

[28] J. Sola and J. Sevilla, “Importance of input data normalization for the application of neural networks to complex industrial problems,” *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3 PART 3, pp. 1464–1468, 1997, doi: 10.1109/23.589532.

[29] S. and Y. H. M. Abu Ghurah1, 2, M. K. A. Kamarudin1,*, N. A. Wahab1, R. Umar1, N. A. F. Nik Wan1, H. Juahir1, M. B. Gasim1, A. R. Hassan1, F. Lananan1, A. F. Ireana Yusra1, “Special issue. Special issue,” *J. Fundam. Appl. Sci.*, vol. 4, no. 1, pp. 9–10, 2018.

[30] V. de M. Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, “Detection of reduction-of-quality DDoS attacks using Fuzzy Logic and machine learning algorithms,” *Comput. Networks*, vol. 186, no. December 2020, p. 107792, 2021, doi: 10.1016/j.comnet.2020.107792.

[31] X. Zhang and M. Wang, “Improved SVM classification algorithm

based on KFCM and LDA," *J. Phys. Conf. Ser.*, vol. 1693, no. 1, 2020, doi: 10.1088/1742-6596/1693/1/012107.

[32] W. Liu, S. Liang, and X. Qin, "Weighted p-norm distance t kernel SVM classification algorithm based on improved polarization," *Sci. Rep.*, vol. 12, no. 1, pp. 1–17, 2022, doi: 10.1038/s41598-022-09766-w.

[33] V. Jakkula, "Tutorial on Support Vector Machine (SVM)," *Sch. EECS, Washingt. State Univ.*, pp. 1–13, 2011.

[34] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[35] C. Sciences, "M Asters D Issertation Machine Learning for Malware," no. January, pp. 1–61, 2016, doi: 10.13140/RG.2.2.18107.00801.

[36] K. M. Ting, "Confusion Matrix," *Encycl. Mach. Learn. Data Min.*, no. October, pp. 260–260, 2017, doi: 10.1007/978-1-4899-7687-1_50.

[37] A. Alghazali and Z. Hanoosh, "Using a Hybrid Algorithm with Intrusion Detection System based on Hierarchical Deep Learning for Smart Meter Communication Network," *Webology*, vol. 19, no. 1, pp. 3850–3865, 2022, doi: 10.14704/web/v19i1/web19253.

[38] E. Olyaie, H. Banejad, and M. Heydari, "Estimating Discharge Coefficient of PK-Weir Under Subcritical Conditions Based on High-Accuracy Machine Learning Approaches," *Iran. J. Sci. Technol. - Trans. Civ. Eng.*, vol. 43, no. August, pp. 89–101, 2019, doi: 10.1007/s40996-018-0150-z.

[39] A. Gadhave, "Gold Price Prediction using Machine Learning," *Interantional J. Sci. Res. Eng. Manag.*, vol. 06, no. 05, 2022, doi: 10.55041/ijsrem15027.

[40] S. Sharma, C. Rama Krishna, and S. K. Sahay, “Detection of advanced malware by machine learning techniques,” *Adv. Intell. Syst. Comput.*, vol. 742, pp. 333–342, 2019, doi: 10.1007/978-981-13-0589-4_31.

[41] J. Carlos Fernandez Godinho, P. Miguel dos Santos Alves Madeira Adão, and P. Romano Supervisor, “Malware Detection via Machine Learning Information Systems and Computer Engineering Examination Committee,” no. June, 2018.

[42] M. N. Boussiala, “Test Machine Learing with the Data FISH Test Machine Learing with the Data FISH Reading or Importing Data ” no. August, 2021, doi: 10.13140/RG.2.2.34229.81124.

[43] M. Vakili, M. Ghamsari, and M. Rezaei, “Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification,” 2020.

[44] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, “Machine learning algorithms to detect DDoS attacks in SDN,” *Concurr. Comput. Pract. Exp.*, vol. 32, no. 16, pp. 1–14, 2020, doi: 10.1002/cpe.5402.

[45] J. Li, “Detection of Ddos Attacks Based on Dense Neural Networks, Autoencoders and Pearson Correlation Coefficient,” no. April, p. 89, 2020.

الخلاصة

مع استمرار تزايد عدد هجمات البرامج الضارة، هناك حاجة متزايدة لتقنيات فعالة وكفؤة للكشف عن البرامج الضارة. ظهرت خوارزميات التعلم الآلي الديناميكية كطريقة واعدة للكشف عن البرامج الضارة في الوقت الفعلي. تستفيد هذه الخوارزميات من السلوك الديناميكي للبرامج الضارة لتمييزها عن البرامج الحميدة.

تستخدم هذه الأطروحة مجموعات بيانات DDoS 2019 وتحليل البرامج الضارة المتاحة للجمهور وقاعدة بيانات Microsoft Malware Classification (BIG 2015) وتستخدم أكثر من طريقة لتحديد الميزة في مجموعة البيانات مثل اختيار ميزة الشجرة الإضافية واختيار ميزة الارتباط. وبعد تقييم فعالية أشجار القرار (DT)، والشبكات العصبية Feedforward (FFNN)، وأجهزة المتجهات الداعمة (SVM) لاكتشاف أنواع متعددة من البرامج الضارة بناءً على أدائها في مجموعات البيانات هذه، تم استخدام أسلوب التصويت الصعب لاختيار أفضل خوارزمية تصنيف توفر أعلى معدل تنبؤي لجميع أنواع اكتشاف البرامج الضارة، وذلك لإنشاء نظام تصنيف اختيار ديناميكي، وقد حقق هذا الاقتراح نسبة نجاح بلغت 99.99%.

حقق نهج شجرة القرار نسبة نجاح 100%، وحقق نهج FFNN نسبة نجاح 99.99%، وحققت تقنية SVM نسبة نجاح 99.52%، وفقًا لمجموعة بيانات DDoS 2019. أظهرت مجموعة البيانات الأكبر لعام 2015 أن طريقة DT حققت معدل نجاح قدره 98.17%، وحقق نهج FFNN معدل نجاح قدره 96.89%، وكانت تقنية SVM تتمتع بدقة 93.20%. بالنسبة لمجموعة بيانات تحليل البرامج الضارة، حقق نهج DT معدل نجاح قدره 90.77%، وحقق نهج FFNN معدل نجاح قدره 88.88%، وحقق أسلوب SVM معدل نجاح قدره 80.0% فقط.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كليه تكنولوجيا المعلومات – قسم البرمجيات

خوارزمية تعلم الآلة الديناميكية لاكتشاف البرامج الضارة المتعددة

رسالة مقدمه الى
مجلس كلية تكنولوجيا المعلومات ـ جامعة بابل كجزء من متطلبات
نيل درجة الماجستير في كلية تكنولوجيا المعلومات / البرمجيات

من قبل

زهراء نجاح فاضل لفته

بأشراف

أ. د. وسام سمير بهيه

1445هـ 2023م