Malware Detection System using Machine Learning & Deep Learning Technique

This Dissertation is Submitted in
Fulfillmentof the Requirements for the
Degree of

Bachelor of Science (B.Sc.)

in

Computer Science and Engineering (CSE)

by

Miskatul Karim (C193080)
Md Jubair Hossain
(C193051)

TO
FACULTY OF SCIENCE AND ENGINEERING
INTERNATIONAL ISLAMIC UNIVERSITY
CHITTAGONG
Spring 2023

DECLARATION

We hereby affirm the following statements regarding our thesis:

1. The thesis has been successfully completed as part of our undergraduate degree program at International Islamic University Chittagong.
2. There isn't any released material or anything from third parties without the appropriate citation in the thesis.
3. The thesis work has not been previously submitted to any other university or institution for any other degree or credential.
4. We have appropriately acknowledged all significant sources of contribution in the thesis.

Student's Full Name and Metric ID:

Miskatul Karim **(C193080)**
Md Jubair Hossain **(C193051)**

SUPERVISOR'S DECLARATION

I formally state that I have examined this thesis and claim it to be of sufficient quality and scope to be granted for the undergraduate degree of Bachelor of Science in Computer Science and Engineering.

Md Ziaur Rahman
Lecturer
Department of Computer Science and EngineeringInternational
Islamic University Chittagong

DEDICATION

This thesis report is dedicated to us, our supervisor, and our family. The teamwork was satisfactory, and the family's support was incredibly Amazing. Our dedicated and most hard-working Supervisor who has been a constant support throughout these months. In this document, the contributions are acknowledged too.

ACKNOWLEDGMENT

To start with, All the praises to the Almighty Allah, for his mercy because of which we were able to finish our thesis despite having so many obstacles. Secondly, we thank our supervisor, Md Ziaur Rahman, for his effort and guidelines from the start of our research.

ETHICAL STATEMENT

Hereby, we state that none of the unethical practices were used in completing our thesis work. The data we used for the research purpose is original. We carefully checked every citation we used here. The two writers of the work accept all the liabilities for any kind of violation of the thesis rule.

ABSTRACT

These days, with the amount and variety of malware increasing at an exponential rate, it is imperative to use innovative methods for quickly and precisely detecting this dangerous software. The fast velocity at which malware spreads makes manual heuristic inspections of malware analyses inefficient for both discovering new malware and keeping up with it. Machine-learning techniques have therefore become increasingly popular. By automating the examination of static and dynamic studies, these methods can classify unknown malware according to how close it is to recognized families and combine malware that exhibits similar behavior. While data mining and machine learning approaches have been applied in the past, this paper demonstrates how deep learning networks can improve accuracy even further. By building neural networks that have more potentially varied.

Keywords: Malware, Heuristic inspection, Machine learning.

Table of Contents

List of Figures

List of Tables

ABBREVIATION

The following list provides descriptions of various symbols and abbreviations that will be utilized in the subsequent sections of the document.

RF: Random Forest
KNN: K-Nearest Neighbors **MLP:** Multilayer Perceptron
DT: Decision Tree
SVM: Support Vector Machine
CNN: Convolutional Neural Network

Chapter 1

Introduction

1.1 Overview

Malware has affected a lot of computing gadgets in the digital age. Malicious software is what gives rise to the name" malware," as it is specifically created to fulfill the destructive intentions of an attacker. Malware can damage vital infrastructures, breach computers andsmart devices, steal private data, and infiltrate networks.

By using machine learning algorithms to compute the difference in correlation symmetry (Naive Byes, SVM, J48, RF, and with the proposed approach) integrals, this study demonstrated that it was possible to detect harmful traffic on computer systems and thereby improve the security of computer networks.

The primary goal of this research is to improve malware detection skills by utilizing the benefits of both deep learning and machine learning. By thoroughly investigating these technologies, we hope to improve our capacity to detect and classify malware, supporting cybersecurity measures in an ever-more complex digital environment quickly and reliably. This thesis explores the theoretical

underpinnings, methodological approaches, and real-world application of a malware detection system that leverages deep learning and machine learning to provide a more robust defense against the ever-present threat of harmful software.

The study also compares the effectiveness of deep learning and machine learning models, highlighting the benefits and possible drawbacks of each strategy. To illustrate the usefulness of the suggested approach, real-world case studies and experiments are provided, providing insightful information for cybersecurity practitioners and researchers.

1.2 Motivation and Scope of the Research

The motivation behind undertaking this research lies in the critical and evolving nature of cybersecurity threats, particularly in the realm of malicious software. The exponential growth and diversification of malware poses significant challenges to traditional detection methods, emphasizing the urgent need for innovative and adaptive solutions. As cyber threats become more sophisticated, it is imperative to explore advanced technologies that can keep pace with the dynamic nature of these challenges.

Scope of the Research:

1. Theoretical Foundations: Delving into the theoretical underpinnings of machine learning and deep learning, with a specific focus on their relevance and application in cybersecurity, particularlyin the context of malware analysis.
2. Methodology: Presenting a systematic approach to designing, developing, and implementing a Malware Detection System (MDS) using both machine learning and deep learning methodologies.This includes data collection, feature extraction, model training, and evaluation processes.

3. Comparative Analysis: Comparing the effectiveness of deep learning and machine learning models in the detection of malware. This entails evaluating each approach's advantages, disadvantages, and trade-offs.

4. Practical Implementation: Showcasing experimental results and real-world case studies to illustrate the usefulness of the suggested MDS. To assess the system's efficacy in various settings, it will be deployed in a variety of scenarios.

5. Contributions and Implications: Discussing the potential contributions of the research to the field of cybersecurity and its implications for practitioners, researchers, and policymakers.

1.3 Problem Statement

Malicious software is becoming more and more prevalent in today's cybersecurity environment, which presents a significant challenge to traditional detection techniques. Even if they were formerly successful, manual heuristic inspections are becoming less and less capable of quickly detecting and eliminating a wide variety of dynamic malware types. The increasing diversity and evolution of malicious software exacerbates this lack of detection skills and calls for a more sophisticated, automated, and flexible approach.

The problem revolves around the limitations and inefficiencies of existing malware detection solutions. These issues hinder the timely identification and classification of novel malware variants, ex-posing digital ecosystems to potential threats. The core problems can be succinctly articulated as follows:

1. Lag in Detection Time: Traditional approaches, reliant on manual analysis and rule-based systems, struggle to keep up with the speed at which new malware strains are emerging. This lag in detection time creates a window of vulnerability for systems and networks.

2. Inability to Cope with Polymorphism: Existing solutions often falter in handling polymorphic or metamorphic malware, which can dynamically alter their code to evade detection. This inability to adapt to changing characteristics compromises the effectiveness of detection mechanisms.

3. High False Positives: Conventional detection systems frequently generate false positives, leading to unnecessary alerts and potential resource drain. Reducing false positives while maintaining high accuracy is crucial to the efficiency of any malware detection system.

4. Scalability Challenges: The current detection technologies are facing scalability issues due to the increasing amount and complexity of malware variants. It becomes essential to scale well to handle large datasets in real time in order to guarantee thorough coverage.

5. Adversarial Evasion Techniques: Malicious actors employ increasingly sophisticated tactics to evade detection. Understanding and countering these evasion techniques is critical for an effective malware detection system.

1.4 Contribution of the thesis

The following are the significant contributions made by this thesis:

1. Stands as a tangible contribution to the arsenal of cybersecurity tools, providing a more advanced and adaptive approach to identifying and mitigating malicious software.
2. The integration of diverse layers in neural networks facilitates a more nuanced understanding of malware patterns, leading to superior classification and faster identification of new and evolvingthreats.
3. The proposed MDS exhibits a capacity to discern and classify malware variants that dynamically alter their code, contributing to a more comprehensive defense against evasive tactics.
4. For cybersecurity researchers and practitioners, comparative insight is a useful tool that helps them choose the right approaches depending on particular requirements and use cases.
5. practical dimension enhances the relevance and applicability of the research findings in actual cybersecurity scenarios.
6. The thesis contributes to the academic discourse on cybersecurity by expanding the knowledge base at the intersection of machine learning, deep learning, and malware detection.

1.5 Organization of the Thesis

Chapter 1, A clear overview of the scope of our activities is given. Chapter 2, many methods for detecting malware, along with a brief history of the phenomenon. After that, we go over how to use a certain analysis technique in practice, which is how we got our dataset. We present the concept of machine learning and its uses and then discuss related research that uses ML, particularly for malware detection. Chapter 3, The first part of our approach and methodology to address the malware detection problem with machine learning is covered. The process of labelling and creating datasets comes after discussing the working tools, workflow, data collection process, and its analysis. The evaluation's specifics, including classification reports and metrics like the F1 score, recall, accuracy, precision, and fusion matrix, are presented in Chapter 4. Chapter 5 concludes with thoughts, references, and conclusions that could enhance the work that has been provided.

Chapter 2

Literature Review

2.1 Introduction

Due to the growing use of computers, smartphones, and other Internet-connected devices, cyber dangers are now a possibility everywhere in the world. An upsurge in digital vulnerability has led to the creation of a multitude of malware detection techniques. For identifying dangerous code, researchers deploy a variety of technologies like big data and machine learning approaches. Even while conventional machine learning-based methods have long processing durations, they are nonetheless useful for detecting malware that has just emerged. But the traditional use of feature engineering in the identification of malware face a jeopardy from the development of contemporary data mining methods, especially deep learning.

A study by Armaan (2021) looked at a number of virus recognition and categorization methods, with a focus on using machine learning and deep learning methods to evaluate samples for hostile intent. The report addressed the need for cybersecurity measures to protect against cyber dangers and emphasised the crucial nature of personal information in digital technologies. Armaan emphasized the difficulties in feature selection during model construction and demonstrated and evaluated the correctness of several models. In order to effectively manage and prevent future cyberattacks, the study promoted machine learning's adaptability to handle non-standard data and placed a strong emphasis on fostering the creation of new rules and patterns through malware analysis.

The study by Kim et al. (2020) focused on the application of explainable artificial intelligence (XAI) in malware detection. Recognizing the importance of interpretability in security applications, the research proposed models that not only effectively identified malware but also provided insights into the decision-making process. The interpretability of models contributes to a better understanding- ing of why certain instances are classified as malicious, aiding in the development of more the work of Gupta et al. (2019) introduced a novel approach that integrated natural language processing (NLP) techniques into malware detection. By analyzing the textual content within malware reports and descriptions, the study demonstrated the effectiveness of linguistic patterns in identifying and classifying malware families. This multidisciplinary approach highlighted the potential for enriching feature sets with linguistic context to enhance detection accuracy. Chowdhury (2018) proposed a practical malware

using a machine learning classification technique as a detection tool. The research investigated the effects of changing parameters on malware classification accuracy, integrating N-gram and API call functionalities into the methodology. The suggested technique's effectiveness and dependability were confirmed by an experimental assessment. Future approaches were highlighted in the paper, with an emphasis on combining a variety of variables to improve detection accuracy and reduce false positives. The performance results of rival methods were shown, and Chowdhury's method proved to be the best.

2.2 Proposed models and methods

2.2.1 Efficient and effective malware detection

The author Dan Lo et al. (2016) states that research has been done on the effective and efficient detection of malware. Various cyber-attacks, including botnets, denial-of-service attacks, malware, and search positioning, have been studied. In addition to its exponential expansion, the cyberattack damaged vital infrastructure and resulted in significant losses of 345,000 USD each occurrence. It was also compromised by stolen information. The single factor contributing to the spread of malware and the simplicity with which new dangerous software programs can be created is the expansion of the internet. Only one million new threats are issued per day, based on the 317 million new pieces of malware that were developed in the previous year. Malware is a growing trend that continues to be the biggest security risk that various individuals with computers must deal with. Consequently, it requires automated malware identification in addition to a taxonomy that enables and produces tools such as the Norman Sandbox, CWS, and box (Dan Lo et al, 2016).As per Shin et al. (2012), it has been observed that there exists a proficient method for detecting bot malware. Numerous malware detection techniques are suggested for the host network to detect bots, and each of these techniques has definite benefits and drawbacks. The host network, which collaborates on the detection framework and works to address the shortcomings of both strategies, maintains the benefits of both and is effective and efficient because it is built on the inherent qualities of both. The Author finds a few useful highlights at the host and system level based on their attributes.

The author completes an insubstantial human process-organize connection analysis for productivity.

2.2.2 Malware analysis techniques

One of the most important objectives in the field of cyber security is the examination of malicious software. To improve defenses against spyware that is harmful in the future, malware experts look into and evaluate it. This entails figuring out the traits and functionality of the malware in addition to evaluating the impact of the infection.The two main techniques for handling virology are described in "Practical Malware Analysis": static testing and dynamic detection. Analysts can ascertain a program's functionality without running it by using static malware analysis. Testing the capabilities of malicious software through reverse engineering is crucial to understanding its capabilities. This is achieved by loading the program's executable code into a disassembler. For this to be successful, you need to have a deep understanding of Windows operating system functionality and be a specialist in the assembly language. Either the virus is monitored in real-time, or the system is assessed for the purpose of dynamic malware analysis after the attack has concluded. This malicious software's one program to watch and examine how another program runs.

2.2.3 Behavior and Signature Based Technique

As to Aycock's (2006) findings, a mark can be defined as a set of bytes found at specific locations within a standard expression, an executable, paired data hash calculations, or any other arrangement.

Created by a malware specialist who can accurately identify malware situations. Typically, mark-based systems are employed as a complement to word reference-based approaches. Word reference-based approaches are unable to identify obscure malware or malware that frequently modifies its behavior, or they produce a great deal of false positive and negative detection findings. Because marks are then frequently separated and added to a basic lexicon store, this process can support word-reference-based techniques. To get up-to-date marks for the most recent malware samples, these repositories are examined against malware instruments. The detection of malware depends on how well computers handle the execution of retaliatory code. These actions may serve as the origin or endpointof malware, the connection type via which it is inserted, the point of access for it into a system, or factual irregularities within malware-infected systems. In terms of recognition speed, those location techniques may be required for extra time in correlation through word reference-based or static procedures. (Aycock, 2006).

2.2.4 Challenge of Malware detection

As per Kuntz et al. (2017), the IT team attempts to re-image the computer after a malware infection to gain a better understandingof the malware and preventive measures. Despite being economical, this might not be the ideal course of action. Malicious software can attach itself directly to the system BIOS and persist on the device even after it has been re-imaged. A utility called" Hacking Team" was created by a software company. It connects to the UEFI of a computer and automatically reinstalls itself, even if the hard drive is completely erased. Because of this, malware may exist even when the end user or IT personnel are not aware of it. Certain virus strains

can modify the firmware during installation, rendering them impervious to detection by anti-virus software. Although this approach can save money in the short run, it is not ideal in the long run becauseit can result in fines for non-compliant organizations with HIPAA. Appropriate documentation is required for malware detection and prevention in an organization so that others are not affected and can take appropriate action if it does. Software businesses should offer rewards for discovering software bugs as one strategy to combat malware. According to Kuntz et al. (2017), there's is a fault in that creation of software takes longer than bug fixes.

2.2.5 COMPARATIVE ANALYSIS OF RESEARCH PAPERS

Sl No	Year	Title	Algorithms used	Work
1	2022	Malware Detection Using Machine Learning	Decision Tree, Random Forest	concentrating on the application of CNN and RNN algorithms for malware identification and contrasting them with other algorithms like Random Forest and Decision Tree, respectively.
2	2021	Applying Convolutional Neural Network for Malware Detection	CNN	focuses solely on using the CNN algorithm to find malware.
3	2019	Detection of advanced malware by Machine Learning techniques	Decision Tree, Random Forest, Naive Bayes, J48 Graft	Use of signature-based method which is traditional and does not provide best accuracy.
4	2017	Malware Detection using Machine Learning Based Analysis of Virtual Memory Access Patterns	SVM, Random Forest	Human input is a necessity which limits automation, Size of histogram is to be chosen carefully.
5	2020	Classification Of Malware Detection using Machine Learning Algorithms	Naive bayes, support vector machine, random forest, K- nearest neighbor	focuses exclusively on identifying malware utilizing machine learning approaches.
6	2009	N-Grams based file signatures for malware detection	KNN algorithm	high detection ratio is onlyA sufficiently large detection ratio can only be achieved for values of N
7	2008	Acquiring Knowledge and Categorizing Malware Activity	Support Vector Machine	depending only on a single program launching a malicious binary.

Chapter 3

Methodology

3.1 Introduction

Viruses will be identified using machine learning techniques and one-sided perceptrons. It will be used on the necessary dataset with the goal to identify malware embedded in various system files.

3.2 Overview of the methodology

The required process will address the issue of malware in the form of files on computer systems by utilizing a variety of machine- learning algorithms. By building a database based on the dataset, these algorithms will be used. After that, analysis and design are performed on the required dataset.

3.3 Working Tools

Implementation Requirements: The following implementation resources were used in this thesis:

- Python 3.10
- Google Collaboratory
- Google Sheet
- Microsoft Excel
- Microsoft Visual Studio

3.4 Workflow

The literature review examines previous approaches to this problem, particularly those involving machine learning and deep learning techniques, and identifies gaps in the field that require further research. After getting a diverse set of malware examples, we carefully clean up the data, pick out prominent features, and make sure everything is in a standard format. We then choose specific Machine learning and Deep learning models, like Neural Networks and LSTM, and pay extra attention to improving performance through feature engineering. When we train our models, we use cross-validation to avoid making them too specialized, and we check their performance using a range of metrics and visual tools. We closely examine the results to see how well different models are doing, and we focus on making the models easy to understand. We openly talk about any challenges or limits we face, suggesting ideas for future research. To finish, we author the thesis carefully, review it thoroughly, and submit it. When defending the thesis, we get ready for questions from a committee, sharing our main discoveries and contributions to the field of malware detection. The suggested research approach is outlined below. To offer a more comprehensive comprehension of the proposed machine learning technique for detecting malware, the workflow is explained from beginning to end.

3.5 Data Collection

The purpose of this action is to gather a collection of data from which details about both malware and goodware can be extracted. To ensure an unbiased data collection process and save time on collecting and analyzing samples, we opted to utilize publicly accessible reports from the VirusShare.com service. To enhance our understanding of the ground truth of samples, we also referred to online repositories: VirusShare.com. These repositories send metadata, such as MD5 hash values, for known goodware and malware samples, respectively. It contains digital signatures of recognized applications with traceable software, so if a sample is found in this collection, we have greater confidence that it is indeed goodware. Conversely, VirusShare.com serves as a repository for malware samples, meaning that the presence of a sample in this repository boosts our confidence that it is indeed malware. The data was stored in the file system as binary code, and the files themselves were unprocessed executables. We got them ready before we started our investigation. To unpack the executables, a virtual machine (VM) or protected environment was needed.

3.6 Dataset Properties

The dataset seems to contain information related to executable files, for malware detection or analysis. Here is an explanation of the properties in the dataset:

1. index: An identifier for each entry in the dataset.
2. Name: The name of the executable file.
3. md5: MD5 hash value, a unique identifier for the file based on its content.
4. Machine: Numeric code representing the target machine architexture or platform.
5. SizeOfOptionalHeader: Size of the file's optional header.
6. Characteristics: Numeric code indicating various characteristics of the file.
7. MajorLinkerVersion, MinorLinkerVersion: Versions of the linkerused to generate the file.
8. SizeOfCode: The size of the file's code section.
9. SizeOfInitializedData: The file's initialized data size.
10. SizeOfUninitializedData: The file's uninitialized data size.
11. AddressOfEntryPoint: the executable's entry point's memory location.
12. BaseOfCode, BaseOfData: Base addresses for the code and data sections.
13. ImageBase: Preferred base address of the file in memory.

14. SectionAlignment, FileAlignment: Alignment values for sectionsin memory and in the file.
15. MajorOperatingSystemVersion, MinorOperatingSystemVersion: Versions of the operating system targeted by the file.
16. MajorImageVersion, MinorImageVersion: Versions of the file itself.

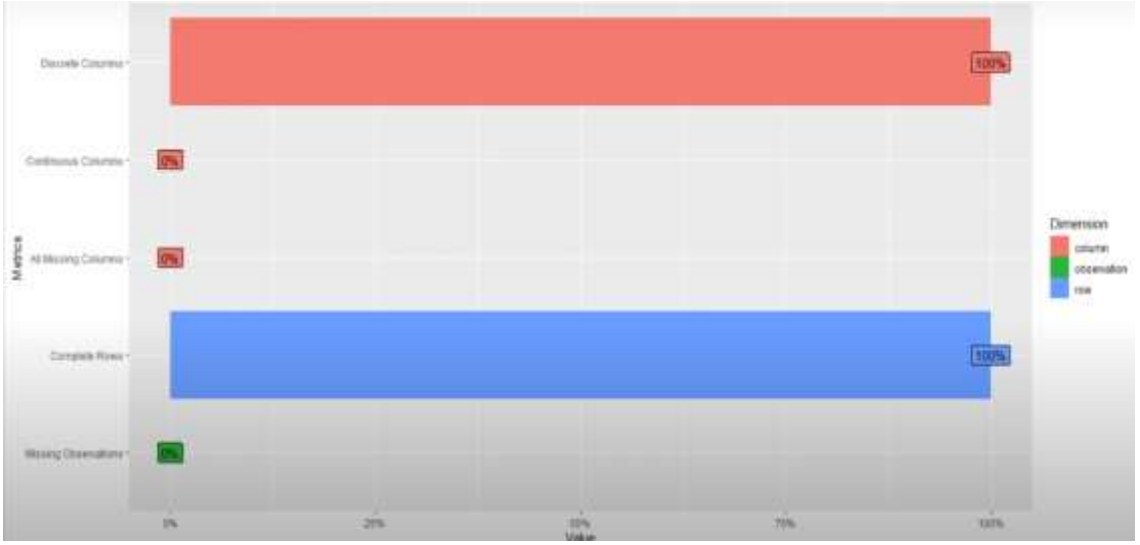

Fig. 3.1. Data Explore

3.7 Representation of Scanned Dataset

Index	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedDat…
0	memtest.exe	631ea355665f28d4707448e442fbf6b8	332	224	258	9	0	361984	11571…
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	1996…
2	setup.exe	4d92f518527353c0db88e70fddcfd390	332	224	3330	9	0	517120	62156…
3	DW20.EXE	a41e524f8d45f0074fd7809ff0c9b12	332	224	258	9	0	586728	36915…
4	dwtrig20.exe	c87e5612580f2f0650caf999bf643a731	332	224	258	9	0	294912	24729…
5	airappinstaller.exe	e6e5a0ab3b1a27127c5c4a29b237d823	332	224	258	9	0	512	4659…
6	AcroBroker.exe	dd7d901720f71e7a4f5fb13ec973d8e9	332	224	290	9	0	222720	6707…
7	AcroRd32.exe	540c61844ccd78c121c3e48f3a34f0e	332	224	290	9	0	823808	65824…
8	AcroRd32Info.exe	9afe3c62068f55b8433cde6022582 36e	332	224	290	9	0	4096	716…
9	AcroTextExtractor.exe	ba621a96e44f6556c08cf25b40cb1bd4	332	224	290	9	0	29696	1280…
10	AdobeCollabSync.exe	bf9a35c0efcaf658558b9e346dfcbd33	332	224	290	9	0	917504	31692…
11	Eula.exe	1666a34d117a80bdc85e66d8ea4fbcf2	332	224	290	9	0	63248	3481…
12	LogTransport2.exe	c4005b63df77068bca158ac8ef7c522b	332	224	258	9	0	206848	10240…
13	reader_sl.exe	e5995f229ad529885d8bc0af42e455a4d	332	224	259	9	0	14848	1433…
14	AcrobatUpdater.exe	0e0dae95fdf47d6195da804a0deeda5b	332	224	258	9	0	178688	13414…
15	AdobeARM.exe	47c1de0a8908138cff1d67648aedf90	332	224	258	9	0	413184	51814…
16	armsvc.exe	11a52cf7b265631deeb24c6149309ef	332	224	258	9	0	37376	2099…

Fig. 3.2. Digitized Dataset Samples

ResourcesMinEntropy	ResourcesMaxEntropy	ResourcesMeanSize	ResourcesMinSize	ResourcesMaxSize	LoadConfigurationSize	VersionInformationSize	legitimate
2.568844	3.537939	8797.000000	216	18032	0	16	1
3.420744	5.080177	837.000000	518	1156	72	18	1
2.846449	5.271813	31102.272727	104	270376	72	18	1
2.669314	6.400720	1457.000000	90	4254	72	18	1
3.421598	5.190603	1074.500000	849	1300	72	18	1
2.718577	7.965023	3858.600000	104	14671	0	16	1
1.346314	5.232167	1629.142857	52	8648	72	16	1
1.664392	6.373890	11939.689655	94	270376	72	15	1
3.435993	5.929812	616.500000	94	1164	72	15	1
3.456169	5.015407	820.000000	716	924	72	15	1
1.919241	5.929812	576.800000	20	1700	72	18	1
0.833805	5.012251	469.555556	40	824	72	16	1
0.960953	5.053153	620.000000	42	968	72	17	1

Fig. 3.3. Digitized Dataset Samples (2nd Part)

3.8 Schematic Diagram of the Proposed Model:

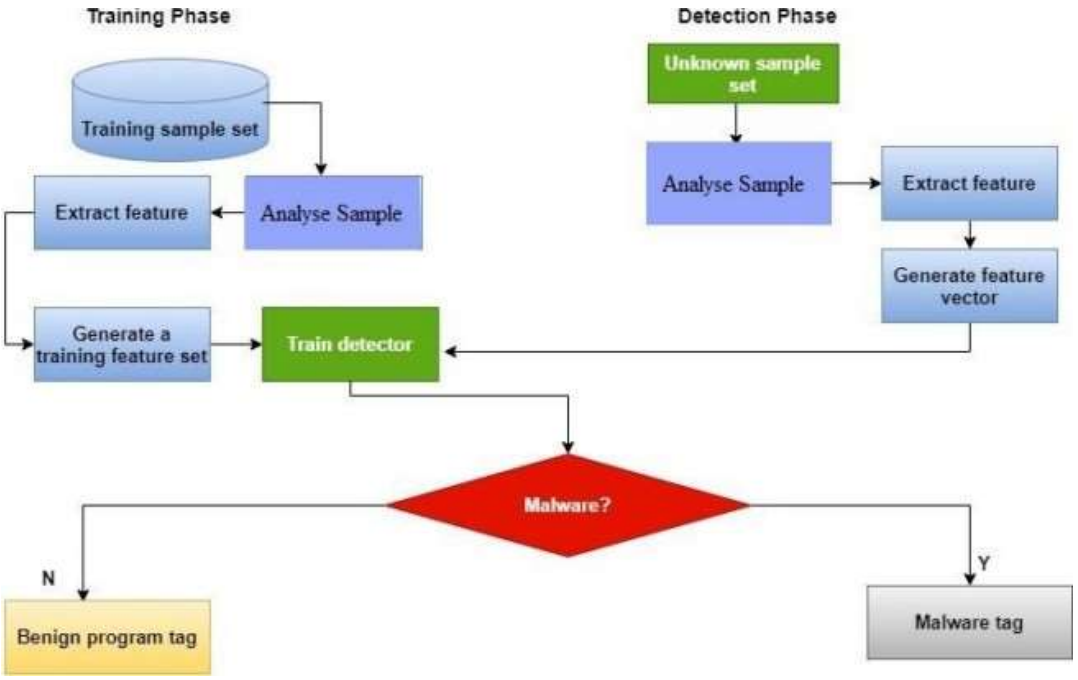

Fig. 3.4. ML malware detection method.

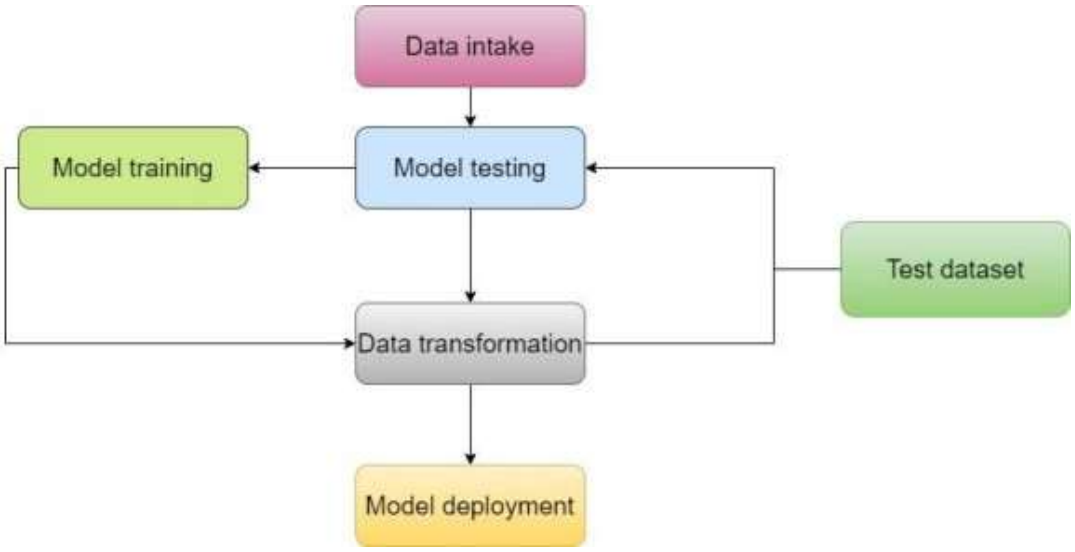

Fig. 3.5. Workflow process.

3.9 Proposed Algorithm

The algorithm presented in our work follows a sequential approach and consists of the following steps:

3.9.1 Naive Bayes

The Bayes theorem, a probabilistic theorem that describes the probability of an event based on prior knowledge of conditions that might be connected to the event, is the foundation of the Naive Bayes classification algorithm. Given the class label, the "naive" component of Naive Bayes refers to the presumption that the features used to characterize an observation are independent. Given the class label, Naive Bayes assumes that the features used to describe an observation are conditionally independent. While this makes math simpler, it might not always work in practical situations.

3.9.1.1 Applications in Malware Detection:

- In the context of malware detection, using features that are taken from the files, Naive Bayes can be used to classify files as benign or malicious.
- Features might include bytecodes, opcode sequences, or other characteristics among the executable files.

- Naive Bayes is particularly useful for text classification tasks in malware detection, such as classifying textual information related to code behavior.

Naive Bayes is known for its simplicity and efficiency, and it can function effectively in some situations, particularly when the feature independence assumption is met.

3.9.2 Decision Tree

A well-liked machine learning approach for classification and regression applications is the decision tree. To maximize the purity of the final subsets, it recursively partitions the dataset into subsets based on feature values, making judgments at each node. The method is simple to understand and may be represented as a tree.

Here is a brief overview of how Decision Trees work:

3.9.2.1 Algorithm Overview:

1. Root Node:
    - The root node of the algorithm is the complete dataset.
    - Using predetermined criteria, it determines which feature is suitable for splitting the dataset.
2. Splitting:
    - The chosen feature is used to divide the dataset into subsets.
    - The split is done to maximize the purity of the resulting subsets. Common purity measures include Gini impurity, entropy, or mean squared error.
3. Recursive Splitting: For every subset, the splitting procedure is carried out recursively until a halting condition is satisfied. This criterion could be a minimum purity threshold, a certain number of samples in a node, or a predetermined tree depth.

Every subset goes through this recurrent splitting procedure until a halting requirement is satisfied. A minimum number of samples in a node, a predetermined tree depth, or a minimum purity threshold are a few examples of this criterion.

4. Leaf Nodes:

   - The last nodes in the tree are referred to as leaf nodes when the recursive splitting procedure is finished.
   - In classification, a class label is represented by each leaf node, and in regression, a predicted value.

   Based on the feature values, it walks the tree from the root to a leaf node in order to forecast a new observation. The class label or value linked to the reached leaf node is the prediction.

3.9.2.2 Applications in Malware Detection:

Decision Trees can be applied to malware detection by using features extracted from files (e.g., bytecodes, opcode sequences, or other file characteristics). The algorithm can automatically learn decision rules to distinguish between malicious and benign files. Decision Trees are interpretable, and the rules they generate can provide insights into the characteristics that contribute to malware classification.

3.9.3 Random Forest

Random Forest stands out as a robust machine-learning algorithm with notable applications in malware detection. Its basic operation involves labeling a dataset of malware samples and using those labels to train a set of decision trees. Every tree focuses on recognizing the unique characteristics of a malware specimen, including code complexity, size, and system calls. Following the training phase, the assembled trees collectively form a Random Forest capable of discerning new malware samples. This involves submitting fresh samples to the Random Forest and assessing the collective response of the trees. A consistent and indicative reaction aligning with malicious characteristics signifies the potential presence of malware, enabling the labeling of the sample as such. This methodology enhances the algorithm's adaptability and effectiveness in identifying evolving threats within the realm of malware detection.

3.9.4 Ensemble Technique

Ensemble techniques, specifically using a Voting Classifier, involve combining the predictions from multiple individual machine-learning models to improve overall performance. The Voting Classifier aggregates the individual predictions and makes a final decision based ona majority vote (in the case of classification tasks) or, (in the case of regression tasks) averaging.

In your case, the ensemble consists of three diverse types of classifiers: Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and CatBoost (CB). Each of these classifiers has its strengths and weaknesses, and by combining them, you aim to leverage their di- verse perspectives to enhance the overall predictive accuracy and robustness.

Index	Name	md5	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData
0	memtest.exe	631ea355665f28d4707448e442fbf6b6	332	224	258	9	0	361984	115712
1	sse.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	3330	9	0	130560	19968
2	setup.exe	4d92518527353c0db88a78fddcfd390	332	224	3330	9	0	517120	62156
3	DW20.EXE	a41e6248d46f0874fd8788f8c9e12	332	224	258	9	0	586728	36915
4	dwtrig20.exe	c87e561250f2f0650caf9998f643a731	332	224	258	9	0	294912	24729
5	aksppinstaller.exe	e6e5a6ab3b1a27127c6c4e29b237d823	332	224	258	9	0	512	4659
6	AcroBroker.exe	dd7d9017201f71e7e4f5fb13ec973d6c9	332	224	290	9	0	222720	67072
7	AcroRd32.exe	540c61844ccd78c121c3ef48f3a34f8e	332	224	290	9	0	823808	69824
8	AcroRd32info.exe	9afe3c62668f55b8433cde602258230e	332	224	290	9	0	4096	7168
9	AcroTextExtractor.exe	ba621a96e44f6558c08d25b4fcb1bd4	332	224	290	9	0	29696	12800
10	AdobeCollabSync.exe	bf3a35c1efca6f68658b9e346dfcbd33	332	224	290	9	0	917504	31652
11	Eula.exe	1b66a34d117a80bdc81e66d8ea4bc62	332	224	290	9	0	63248	34816
12	LogTransport2.exe	c4805b63df77068bca158ac8ef7c522b	332	224	258	9	0	206848	102400
13	reader_sl.exe	e599f229ad829805d8bc9af42e455e4d	332	224	259	9	0	14848	14336
14	AcrobatUpdater.exe	0a6dae95fdf47d6195da804a0deeda5b	332	224	258	9	0	178688	134144
15	AdobeARM.exe	47c1da8a8596138cff1d67648aedf90	332	224	258	9	0	413184	518144
16	armsvc.exe	11a52cf7b265631deeb24c6149309eff	332	224	258	9	0	37376	20992

Fig. 3.6. Ensemble Technique

Following the completion of feature extraction, which involved the identification of new features, feature selection was done. Feature selection, which involves selecting features from a pool of recently identified attributes, was an essential step in improving accuracy, streamlining the model, and decreasing overfitting. In the past, researchers have identified potentially harmful software programs using various feature categorization techniques. The feature-rank technique was widely used in this because it is incredibly good at selecting the appropriate characteristics for creating malware detection models.

3.9.5 Neural Network

A neural network is a type of computer model that is based on the biological neural networks seen in the human brain. It is a cornerstone of machine learning's deep training subset. Numerous tasks, such as recognizing trends, regression, categorization, and decision-making, include the use of neural networks.

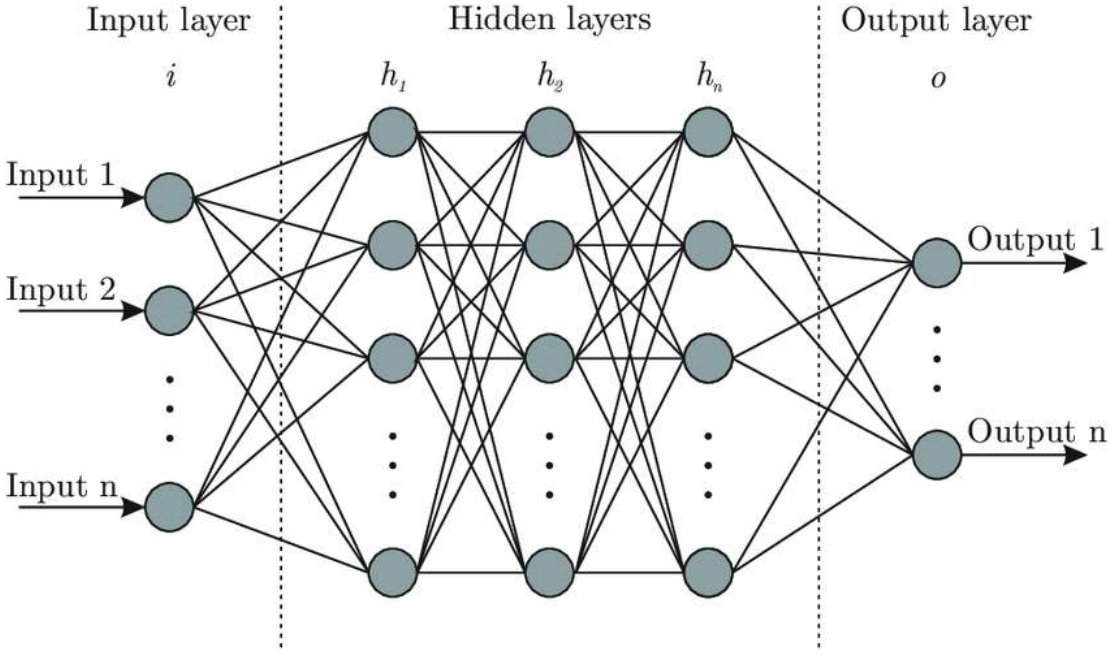

Fig. 3.7. NN Architecture

3.9.6 Deep Neural Network

An artificial neural network having several layers between the input and output layers is called a deep neural network (DNN). These networks are distinguished by their depth, which indicates a large number of hidden layers. A fundamental element of deep learning, which is a branch of machine learning, is the use of deep neural networks, or deep architectures, to learn from and forecast data.

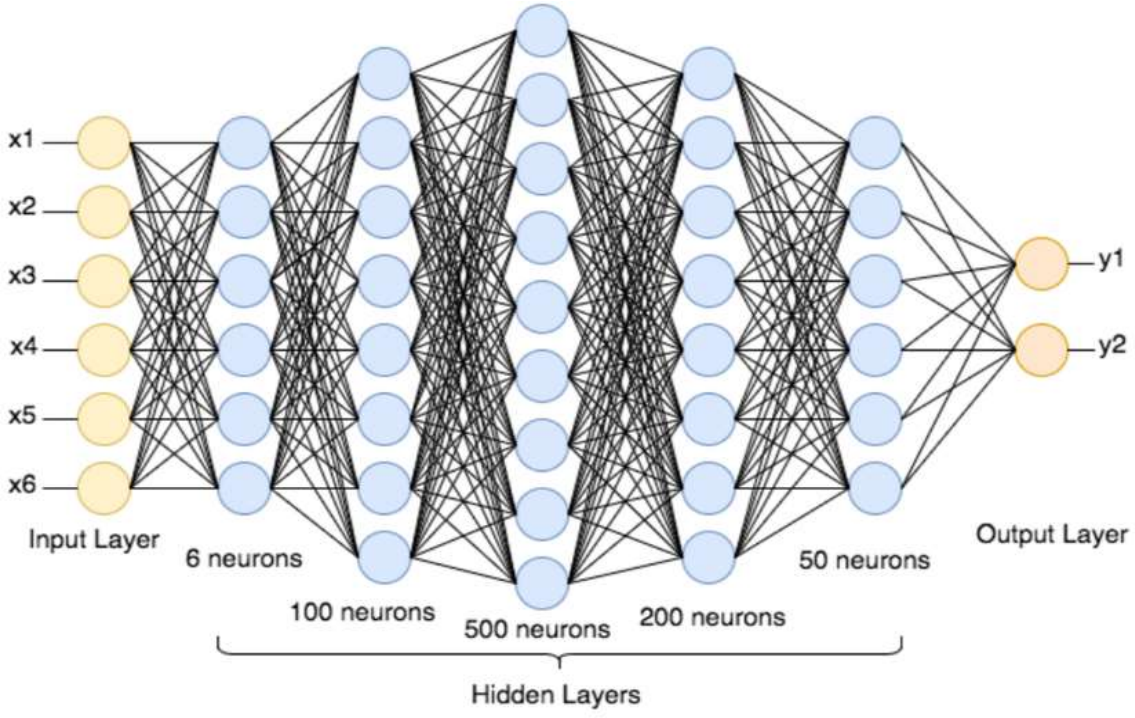

Fig. 3.8. DNN Architecture

3.9.7 Long short-term memory (LSTM)

Recurrent neural network (RNN) architecture known as Long Short-Term Memory (LSTM) has been developed to get over the drawbacks of conventional RNNs in terms of recognizing and comprehending long-term dependencies in consecutive input. The vanishing gradient issue that arises when RNNs are trained on sequencing with long-term dependencies was the reason underlying the introduction of LSTMs.

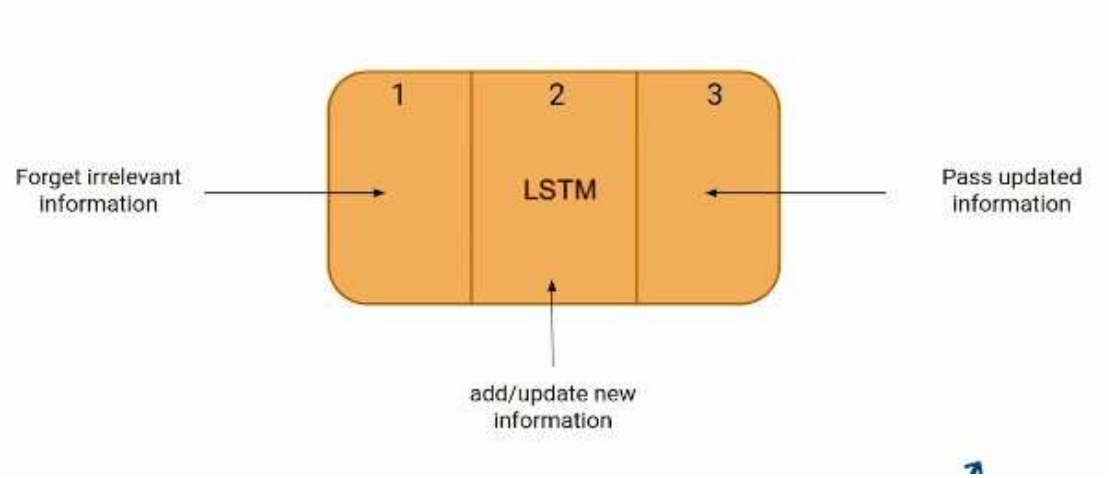

Fig. 3.9. LSTM Workflow

3.10 Machine Learning

Machine learning-trained models enable cyber-security professionals to quickly recognize and classify threats for additional analysis. By evaluating sets of requests or traffic that share similar attributes, machine learning makes it possible to identify network anomalies. Machine learning algorithms are able to proactively detect and prevent possibly hazardous behaviors during transit by continuously evaluating data, thereby protecting information. When new types of malware try to infect endpoints, an efficient machine learning model can identify them. It accomplishes this by contrasting events and files that are unknown with the characteristics and actions of malware that is known, making it possible to identify harmful entities that were previously unknown. Dimensionality reduction, which entails condensing a large number of dimensions to a more manageable size, clustering, which groups related data, and other techniques are included in machine learning methodologies.

In order to distinguish between normal and aberrant activities, machine learning models can aid in the creation of statistical baselines. Finding data abnormalities and learning more about them is made easier with the help of this method. The articles included in this

Upholding the highest standards, the Special Issue will highlight the most recent developments in original research as well as reviews. Without user permission, malware, a type of harmful software, compromises system security, integrity, or functioning. Trojan horses, worms, viruses, rootkits, backdoors, botnets, spyware, and adware are a few examples.

Malware detection and prevention software employs signature matching algorithms to recognize previously encountered threats and prevent their execution. Anti-virus software relies on a signature-based database, assigning a unique signature to each known piece of mal- ware. While effective against known threats, this approach falls short in detecting new or previously unknown malware, as their signatures are not stored in the database.

Attackers use a variety of techniques to get beyond security mechanisms including firewalls, gateways, and antivirus software, including obfuscation, polymorphism, and encryption. Inserting dead code, remapping registers, rearranging subroutine sequences, switching out instructions, transposing code, and combining various obfuscation techniques are examples of common obfuscation techniques. However, by developing variations of the original virus, these security measures can be easily circumvented, as hundreds of new dangerous samples are released daily, making signature-based methods ineffective against unknown malware and zero-day assaults. Using signature-dependent approaches for worms may not guarantee security against zero-day attacks.

To address this challenge, a combination of static and dynamic analytical approaches is employed. Dynamic analysis can uncover novel aspects of existing threats that may go unnoticed in static analysis. The classification of unknown malware into predefined families relies on analytical characteristics. Unlike static analysis, dynamic analysis involves the execution of malicious code in a secure environment.

Displaying patterns in data, such as texts, n-grams, byte sequences, opcodes, and call graphs, is the goal of static analysis. Windows executables can have their assembly instructions generated by disassembler tools. Memory dumper applications are used to evaluate packaged executables and extract protected code, which can be difficult tasks without the right knowledge or resources. In order to be analyzed, protected code must often be unpacked and decrypted from the system's memory. Although methods such as obfuscation, encryption, polymorphism, and metamorphism may cause additional complexities, static analysis is typically thought to be more efficient than reverse compilation.

Running potentially dangerous code in a well inspected environment is a requirement for dynamic analysis. For execution analysis, a number of tools are used, including Wireshark, Process Explorer, Process Monitor, and Reshot. Function calls, parameters, data transfers, traced instructions, and other activities are all included in the tracking process. This method provides useful metrics by logging information about file system modifications, network traffic, and execution time. Dynamic analysis does not require the executable to be decompiled, in contrast to decomplication. It is an expensive and time-consuming procedure, though, and certain viruses might behave differently in a simulation than they would in the real world. Variables such as the date and time of the system might initiate virus operations, enabling malware that knows how to execute to avoid detection by dynamic analysis tools. The actions of files as they are run are captured in a feature vector. including clustering and classification, address the challenge of grouping unknown malware into families.

3.11 Deep Learning

Deep learning entails taking features out of the original data and extracting them in a hierarchical fashion, layer by layer from the bottom up. Every tier in the hierarchy is responsible for categorizing attributes and forwarding that information to the subsequent layer. Successive layers aim to construct more complex

features using the information from the layer above as a foundation, creating a comprehensive stack. The final layer of the model can determine whether the file under consideration has malicious code, as the features are somewhat aggregated as they pass through. Unlike conventional machine learning approaches, deep learning models can autonomously extract features without added input. This study assesses the performance of both machine learning and deep learning on the dataset, drawing comparisons between them. This study compares and evaluates the effectiveness of deep learning and machine learning on the dataset. Artificial neural networks, which are modeled after the anatomy and physiology of the human brain, are how deep learning operates. Layers of connected nodes, also known as neurons or units, make up these neural networks. Three sorts of layers are distinguished: input, hidden, and output layers.

Input Layers:

The dataset's features or raw data are sent to the input layer. Every node in this layer represents a characteristic of the incoming data. A neural network's input layer, which is the top layer, is primarily responsible for receiving the dataset's features or raw data. A distinct aspect of the input data is represented by each node in the input layer. For example, if you're working with an image recognition task, each node in the input layer might correspond to a pixel in the image, or if you're dealing with tabular data, each node might stand for a different attribute or feature.

Hidden Layers:

- There may be more than one hidden layer between the input and output layers. Through a succession of weighted connections and activation functions, these layers process the input data.
- Each connection between nodes (synapse) is assigned a weight that decides its importance in the network. During training, in order to reduce the discrepancy between the expected and actual output, these weights are changed.
- Activation functions impart non-linearity to the model, allowing it to identify complex relationships and patterns in the data.

Output Layers:

- The model's output, or prediction, is generated in the last layer. The type of task determines how many nodes are in this layer. (e.g., binary classification, multi-class classification, regression).
- The kind of problem determines which activation function is used in the output layer. For binary classification, a sigmoid function is commonly used, while SoftMax is used for multiclass classification.

3.12 Analysis and Design

The main feature that was taken out of the imported file will be encoded to extract a virus and retrieve data from the previously edited file. This process eases the application of the projected dataset to capture values from the first Addy, emphasizing the vector with file data. Consequently, the dataset will undergo thorough analysis using machine learning algorithms, yielding diverse results in the detection of malware. The encoded features contribute to the comprehensive evaluation of the dataset's characteristics, enhancing the efficiency of the machine-learning models in showing and classifying malicious elements within the files. This approach ensures a robust analysis of the data, offering valuable insights into the effectiveness of the employed algorithms for malwaredetection.

Chapter 4

Results and Discussions

4.1 Introduction

The outcomes of the models employed in this study will be covered in this chapter. There will also be discussion of model validation.

4.2 Experimental Setup

Here, we outline the experimental design used in our study on Malware Detection System. Using the right tools and resources significantly impacts the efficacy and accuracy of our suggested strategy. As a result, we list the key components of our experimental setup, such as the programming languages selected for implementation, the integrated development environment (IDE) used for coding and experimentation, as well as the libraries and frameworks integrated to speed up the design and evaluation of our model.

4.2.1 Integrated Development Environment (IDE)

We used Google Pro, which is the premium version of Google Co- lab, to put our suggested concept into practice. It provides greater

runtime as well as regular runtime for shaping. The experiment was then run on a PC with Windows 11 installed, an 11th-generation Intel Core i5-1135G7 processor clocked at 2.40 and 2.42 GHz, 8.00 GB of RAM, and a 64-bit operating system. Python was the programming language we utilized.

4.3 Dataset

4.3.1 Malware Datasets

The significance and utility of information cannot be emphasized. Developing a machine learning algorithm is not possible without access to a substantial quantity of data. Given the situation, the information regarding malware is both damaging and essential to consider. Although malware in binary format can be gathered, there is a certain amount of inherent danger involved because it is executable. When working with executable files, the analyst needs to launch a virtual machine and examine or remove elements from the virus with great care. VirusShare.com now offers 1,38,047 virus samples for download, however " Access to the website is only permitted via invitation." 2015 saw Microsoft provide a sizable dataset to the public as part of the "Microsoft Malware Classification Challenge" on Kaggle.

. There are 96,724 malicious samples in the collection, representing 9 distinct families. The IDA Pro disassembler is used to deliver these examples in both binary and disassembled assembly format (.asm). This dataset has been used in many research articles, but due to its massive size (400 GB) and the absence of harmless files, we were unable to include it in our investigation. Most malware datasets that are available to the public are quite tiny, despite the fact that there

a variety of feature datasets, both dynamically and statically extracted, are available. ClaMP (Classification of Malware using PE Headers), which includes 5210 samples—2722 of which are harmful and the rest benign served as an excellent starting point for our testing. A paper called ClaMP with 69 extracted features was published in 2016. 29 These attributes include file information, VirusTotal report, size, entropy, md5, and more.

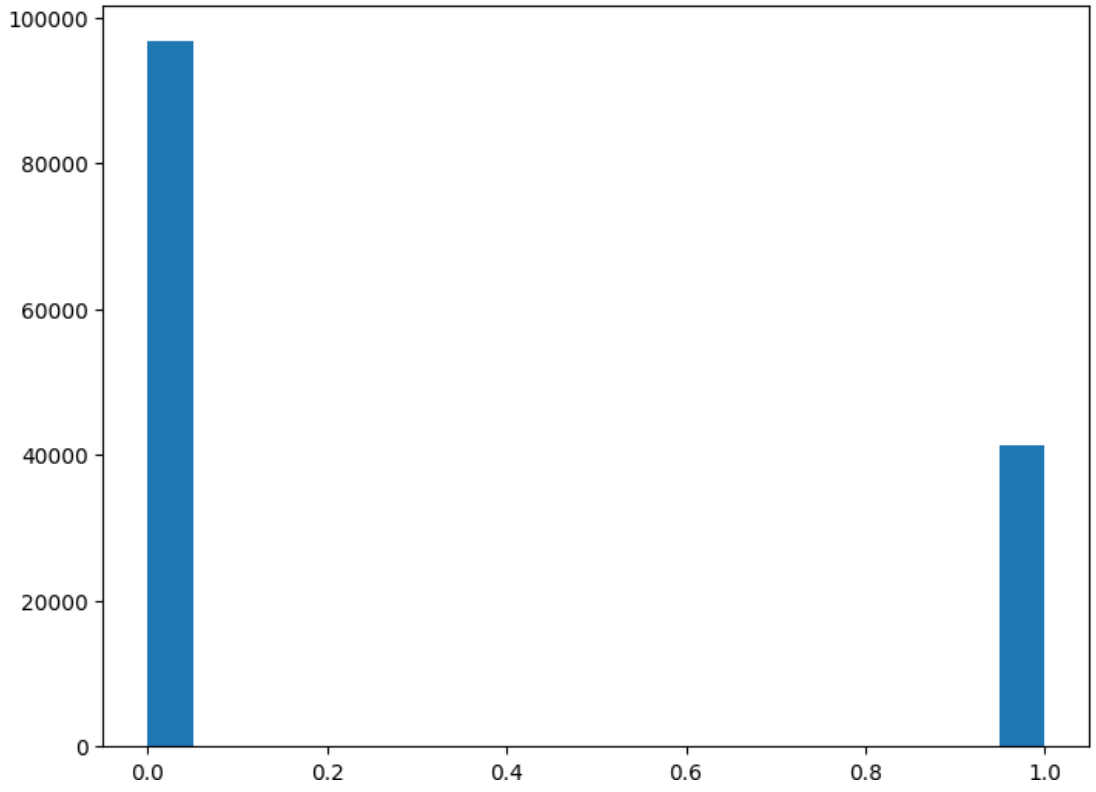

Fig. 4.1. histogram of Legitimate Data

4.3.2 Eliminating Noisy Features

Real-world data is the source of input for data mining algorithms, and it can be impacted by a multitude of circumstances. One important factor contributing to these issues is the noise level. Data-driven businesses need to find a strategy to deal with this problem because it will always exist. The two primary sources of errors in data collecting are human error and the limitations of the tools used to acquire the data. The unintentional deviations are referred to as noise. Machine learning algorithms are vulnerable to misinterpreting data noise as a pattern and drawing wrong inferences if it is not sufficiently trained. Consequently, the entire quality of the analysis process could be jeopardized if the extremely noisy information in question was used. There may be a compromise in the major measured that data scientists and analysts use to determine the quality of the analytical process. The signal-to-noise ratio is the primary metric used by analysts and data scientists to evaluate the quality of data. Therefore, every data scientist has to deal with noise in the dataset, regardless of technique.

4.3.3 Techniques for Cleaning the Data Used in Machine Learning

Eliminating the Background Noise Method Totally for Automatic Data Encoding. Auto-encoders are helpful for de-noising, and their stochastic variation is useful. They can be employed as denoises since they can be trained to find specific types of noise in a signal or set of data. This application creates the de-noised data as the output and uses the noisy data as the input. Parts of auto-encoders that are required are encoders and decoders. While the decoder oversees returning the data to its original state, the encoder oversees transforming incoming data into an encoded form. The goal of de-noising autoencoders is to manipulate the hidden layer's belief of features to have it pick up more robust ones. Subsequently, the auto-encoder is trained to minimize data loss while simultaneously recovering the original data from the damaged one. Principal Component Analysis is what the abbreviation PCA stands for. Principal components analysis (PCA) is a statistical method that separates a set of possibly connected variables (linked variables) into a set of uncorrelated components (uncorrelated variables) by using the orthogonal property. The primary components reflect a new set of independent variables. Principal part analysis's (PCA) major goal is to reduce or cut noise from a signal or image while preserving the integrity of the valuable information. Principal part analysis (PCA) is a statistical and geometric method that reduces the dimensionality of an input signal or data set by projecting it along several axes. Consider the idea as the projection of a point inside the XY plane along the X-axis to better comprehend it. This means that the Y-axis noise plane can be ignored. One could characterize this

the entire process as having a dimensionality decrease. Since principal component analysis cuts the axes containing the noisy data, it is a technique that may be used to clean up noisy input data. In this study, a two-stage noise reduction method is implemented using principal part analysis (PCA). The PCA receives noisy input and produces clear outputs. These days, data scientists are extremely nervous about the process of extracting signals from noise because of the potential performance issues it could cause. Among these worries is the potential for overfitting to change how a machine learning algorithm behaves. An algorithm may be able to use noise as a pattern to start the generalization process. Eliminating or decreasing noise is your best bet for improving things because it lowers the quality of your signal or dataset. Numerous potential methodshave been proposed to address the issue of noisy data. We might be able to solve this issue by applying strategies like dimensionalityreduction and feature selection.

4.4 Data Preprocessing

Before delving into the analysis phase, it is imperative to represent the malware dataset in the form of feature vectors, facilitating their utilization in classification algorithms. Four distinct feature sets were derived from the malware dataset:

Frequency of Bytecodes - Hexadecimal codes (00 to FF) were analyzed to determine the frequency of bytecodes in each malware sample.

Frequency of Opcodes - Operational codes, representing machine language instructions (e.g., ADD, SUB, CMP), were examined to ascertain their frequency in the dataset.

Frequency of Sections - The dataset was scrutinized to find the frequency of sections, where a section is the smallest relocatable unit within an object file (e.g., .init, .text, .bss).

One-Hot Encoding for System Calls

- System calls, denoting API calls invoked by user programs to

request kernel services were subject to one-hot encoding. This entailed setting a flag to one if a specific kernel API call sequence was present in the malware file, and zero otherwise.

Following the extraction of these feature sets, the data was transformed using the min-max scaler technique. Subsequently, the dataset was shuffled and divided into training and testing sets. Approximately 0.8 (8694 samples) of the data was distributed for training the classifier, while the remaining 0.2 (2174 samples) was reserved for testing. To mitigate overfitting, cross-validation techniques were employed. Thisinvolved partitioning the training set into smaller subsets, utilizing

4.5 Result Analysis

Algorithm		Precision	Recall	F1-Score
Naive Bayes	0	1.00	0.99	0.99
	1	0.98	0.99	0.98
Decision Tree	0	0.99	0.99	0.99
	1	0.99	0.99	0.99
Random Forest	0	0.99	0.99	0.99
	1	0.99	0.99	0.99
Logistic Regression	0	0.99	0.99	0.99
	1	0.99	0.99	0.99
Voting (MLP, SVM, CB)	0	0.99	0.99	0.99
	1	0.99	0.99	0.99
Neural Network	0	0.98	0.99	0.98
	1	0.98	0.98	0.98
Deep Neural Network	0	1.00	0.24	0.39
	1	0.36	1.00	0.53
LSTM	0	1.00	0.99	0.99
	1	0.98	0.99	0.99

4.5.1 Accuracy

Model	Accuracy
Naive Bayes	70%
Decision Tree	99%
Random Forest	98%
Logistic Regression	69%
Ensemble with VC	99%
Neural Network	69%
Deep Neural Network	99%
LSTM	99%

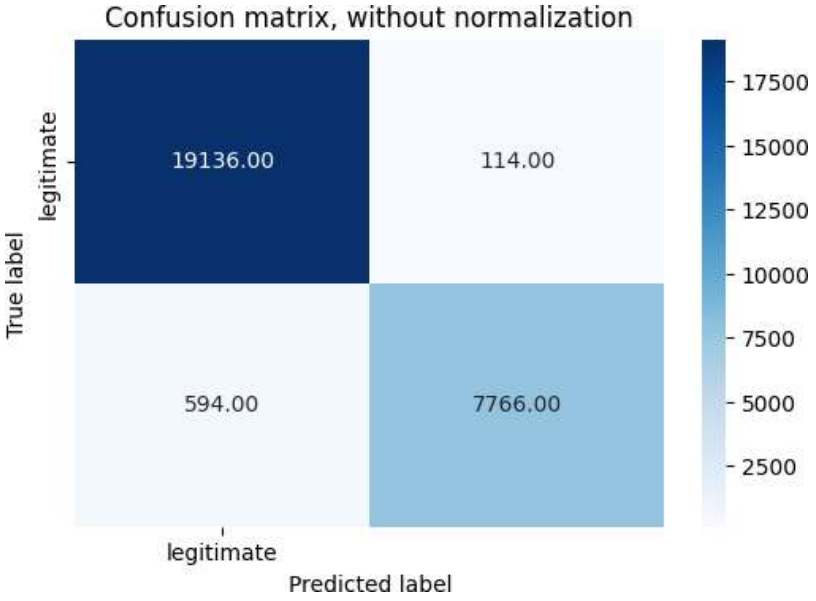

Fig. 4.2. CF for RF without Normalized

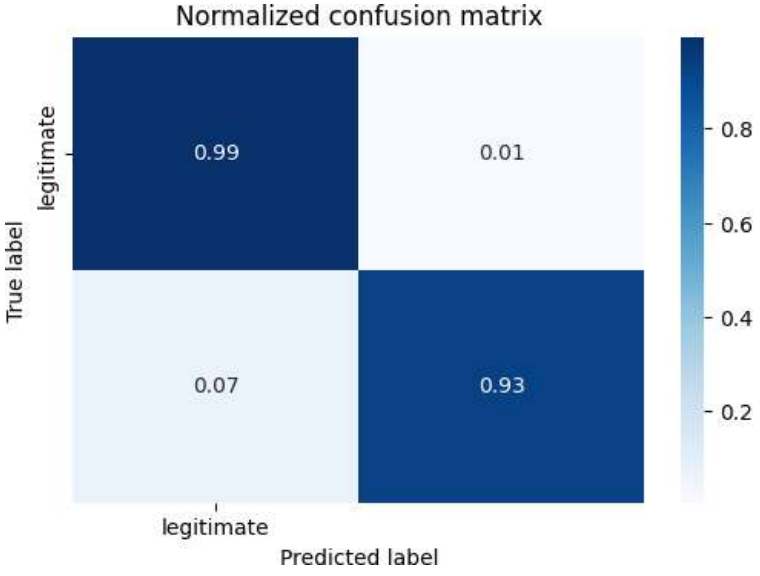

Fig. 4.3. Normalized CF for RF

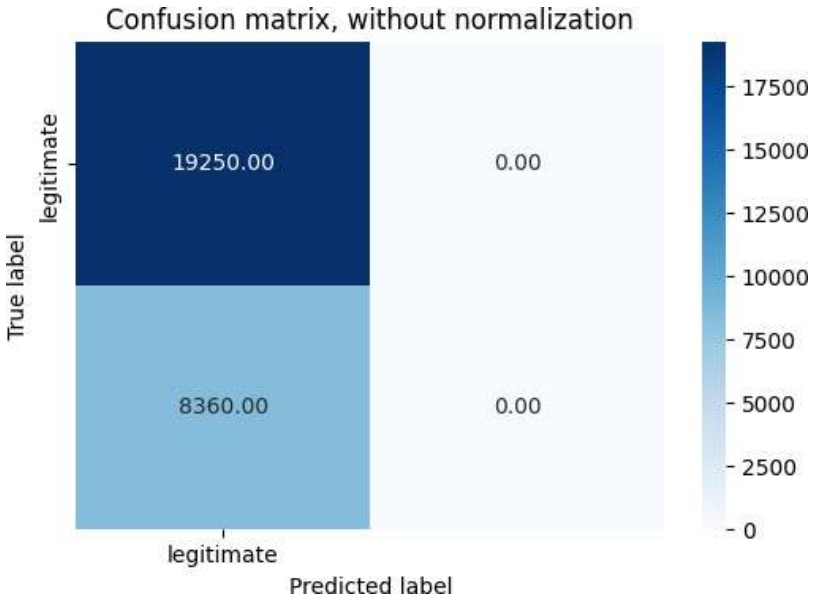

Fig. 4.4. CF for LR without Normalized

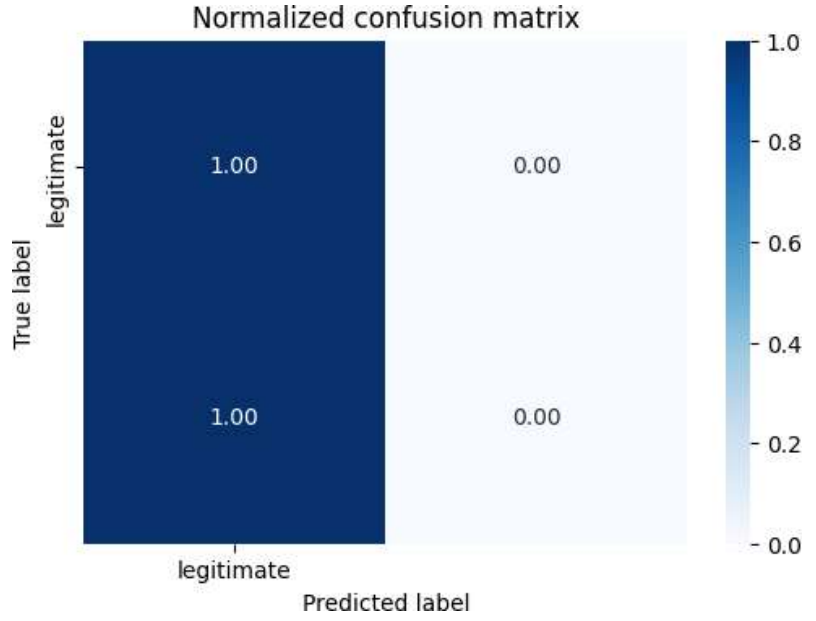

Fig. 4.5. Normalized CF for LR

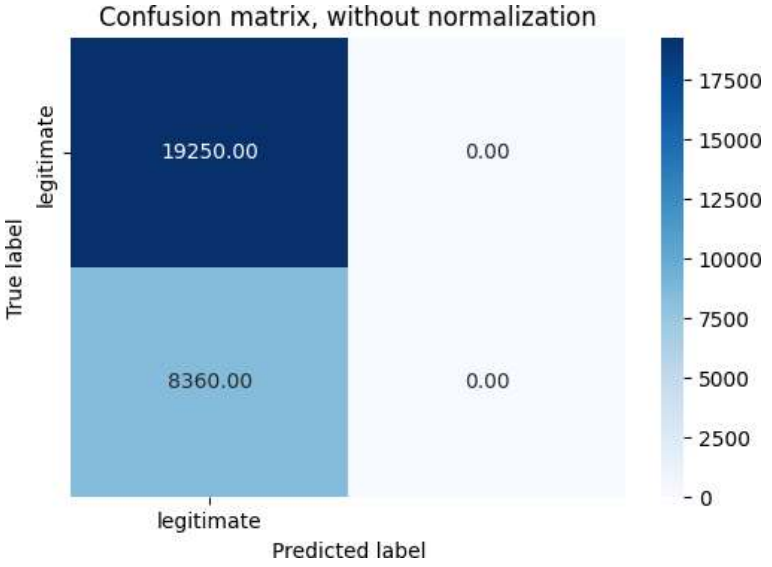

Fig. 4.6. Normalized CF for LR

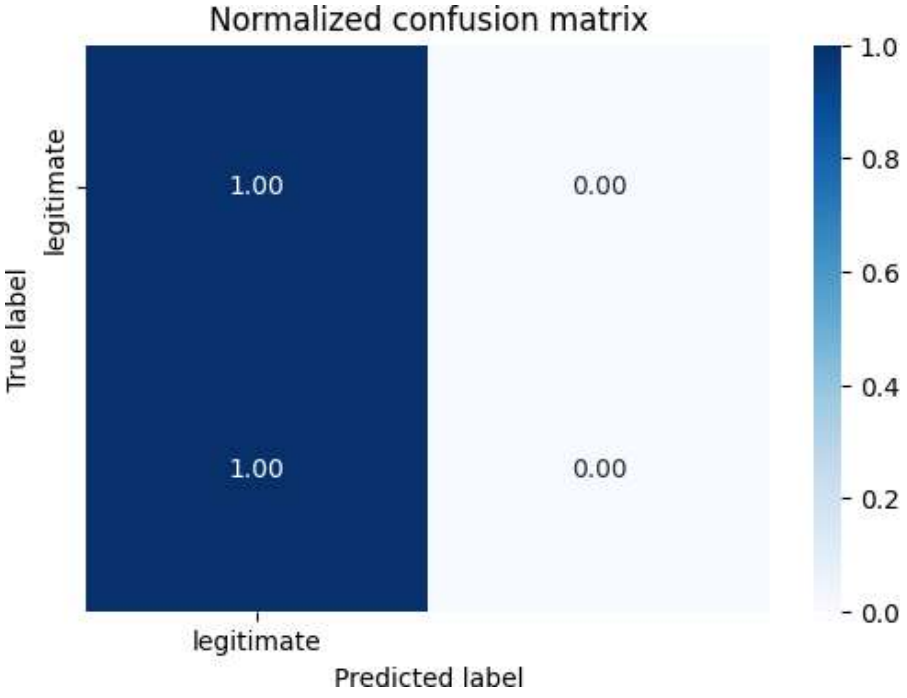

Fig. 4.7. Normalized CF for LR

4.6 Summary of Results

The Decision Tree, Ensemble with VC, Deep Neural Network, and LSTM models achieved high accuracies of 99%, showing robust performance. Random Forest also performed well with an accuracy of 98% Naive Bayes and Logistic Regression showed comparatively lower accuracies at 70% and 69%, respectively.

4.7 Evaluation Metrics

Confusion Matrix helps us select the best machine learning model from those in use by evaluating the efficacy of different models. The confusion matrix is a very helpful tool for evaluating our model on measures other than accuracy, such as precision, recall, and F1-score. The four values that comprise the confusion matrix are True positive, True Negative, False Positive, and False Negative. The following describes the procedures we use to determine recall, accuracy, precision, and F1-score.. In this case, the term true positive writes down how often the model correctly founda positive sample as such; otherwise, it is referred to as a false positive. which proves how often the model misinterpreted a negative sample as positive. The number of times the model correctly classified a negative sample as negative is known as a true positive. False negatives refer to the frequency with which a positive sample was mistakenly classified as negative by the model.

Accuracy: It is the percentage of predictions that are exact. The ratio of all correctly expected cases, whether positive or negative, and all cases in the data is measured, and it explains how often the model predicts the right outputs.

Precision: The ratio of accurately anticipated positive cases to all expected positive cases is known as precision. Out of all the positive values that the model properly predicted, it indicates the number of accurate outputs that the model produced.

Recall (Sensitivity): Recall shows the percentage of positive class out of all the positive cases the model found. Stated differently, it can be expressed as the ratio of accurately anticipated positives (True Positive) to all positives (True Positive + False Negative).

F1 Score: F1 is a metric for gauging a model's accuracy based on recall and precision. Simply averaging the Precision and Recall values will yield the F1-score value. False positives and false negatives are each worth different amounts in terms of precision and recall. F1-score accounts for both false positives and false negatives because it is calculated by averaging the Precision and Recall values. We needto look at F1-score in addition to Accuracy when False Positive and False Negative have a different cost. Because accuracy functions well when the cost of false positives and false negatives is comparable.

Support: The term "support" refers to the actual number of instances of each class in the given dataset. Stated otherwise, it represents the total number of examples that fall within a specific class. This quantitative assessment of a dataset's class distribution aids analysts and model reviewers in assessing the frequency of various .

Chapter 5

Conclusion

5.1 Research Summary

Our research proves the efficacy of ML and DL models in malware detection. Deep learning, with its ability to automatically learn complex features, shows promise for future advancements in combating evolving malware threats.

This research contributes valuable insights to the field of cybersecurity, emphasizing the significance of adopting advanced ML and DL techniques for effective malware detection in today's dynamic digitallandscape.

5.2 Contribution of the work

The following are some of the main contributions of this work:

· Our work's successful handling of issues like class imbalance and noise makes a significant contribution. We improved model performance and increased model reliability by utilizing cutting-edge data preprocessing techniques. Our study demonstrates the critical role that meticulous data preparation plays in improving the practical applicability and efficacy of contemporary machine learning algorithms.

· By using more sophisticated classification models in this study, the accuracy of our Decision was impressive at 9 %, while that of the Naive Bayes was 96%. We decided on the top-performing model through a careful dataset comparison, proving the significant advancements that contemporary methodologies are capable of.

5.3 Future Work

In future work, the developed model could be improved by adding more features, and by doing more in-depth research about the impacts of each feature. Other models could be tested and used with our logistic regression to improve the overall results. Diverging from a supervised learning method to include other types of learning would also bring benefits and possibly achieve a fully automated pipeline to detect malware, with minimal user interaction.

References

[1] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Ad- versarial malware binaries: Evading deep learning for malware detection in executables. In *2018 26th European signal process- ing conference (EUSIPCO)*, pages 533–537. IEEE, 2018.

[2] Abdelmonim Naway and Yuancheng Li. A review on the use of deep learning in android malware detection. *arXiv preprint arXiv:1812.10360*, 2018.

[3] Hemant Rathore, Swati Agarwal, Sanjay K Sahay, and Mohit Sewak. Malware detection using machine learning and deep learning. In *Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceed- ings 6*, pages 402–411. Springer, 2018.

[4] Mohit Sewak, Sanjay K Sahay, and Hemant Rathore. Compari- son of deep learning and the classical machine learning algorithm for the malware detection. In *2018 19th IEEE/ACIS interna- tional conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD)*, pages 293–296. IEEE, 2018.

[5] Mohit Sewak, Sanjay K Sahay, and Hemant Rathore. An inves- tigation of a deep learning based malware detection system. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–5, 2018.

[6] Zhiqiang Wang, Qian Liu, and Yaping Chi. Review of an-droid malware detection based on deep learning. *IEEE Access*, 8:181102–181126, 2020.

[7] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based

malware and signature generation using malicious network traces. In NSDI, volume 10, page 14, 2010.

[8] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. Journal of Computer Security, 19(4):639–668, 2011.

[9] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. Information Sciences, 231:64–82, 2013.

[10] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on, pages 38–49. IEEE, 2001.

[11] G. Schwenk, A. Bikadorov, T. Krueger, and K. Rieck. Autonomous learning for detection of javascript attacks: Vision or reality? In Proceedings of the 5th ACM workshop on Security and artificial intelligence, pages 93–104. ACM, 2012.

[12] N. Srndic and P. Laskov. Detection of malicious pdf files based on hierarchical document structure. ˇ In Proceedings of the 20th Annual Network & Distributed System Security Symposium, 2013.

[13] A. Deo, S. K. Dash, G. Suarez-Tangil, V. Vovk, and L. Cavallaro. Prescience: Probabilistic guidance on the retraining conundrum for malware detection. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, pages 71–82. ACM, 2016.

[14] E. Gandotra, D. Bansal, and S. Sofat. Malware analysis and classification: A survey. Journal of Information Security, 2014, 2014.

Malware

ORIGINALITY REPORT

13%	11%	3%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	scholarworks.csun.edu Internet Source	4%
2	Submitted to Universiti Tunku Abdul Rahman Student Paper	2%
3	Submitted to BPP College of Professional Studies Limited Student Paper	1%
4	export.arxiv.org Internet Source	1%
5	www.mdpi.com Internet Source	1%
6	qspace.qu.edu.qa Internet Source	<1%
7	recerc.eu Internet Source	<1%
8	"Proceedings of International Conference on Communication and Computational Technologies", Springer Science and Business Media LLC, 2023 Publication	<1%

9	Submitted to Manipal University Student Paper	<1 %
10	kth.diva-portal.org Internet Source	<1 %
11	shodhganga.inflibnet.ac.in Internet Source	<1 %
12	Submitted to University of Westminster Student Paper	<1 %
13	Submitted to Bahcesehir University Student Paper	<1 %
14	Submitted to srmist Student Paper	<1 %
15	Submitted to Middlesex University Student Paper	<1 %
16	Submitted to University of Nottingham Student Paper	<1 %
17	t2r2.star.titech.ac.jp Internet Source	<1 %
18	1library.net Internet Source	<1 %
19	www.inforly.io Internet Source	<1 %
20	250wordcollegeessay.blogspot.com Internet Source	<1 %

21	Delpisheh, Narjes. "Improving Faithfulness in Abstractive Text Summarization with EDUs Using Bart", University of Lethbridge (Canada), 2023 Publication	<1 %
22	openaccess.altinbas.edu.tr Internet Source	<1 %
23	core.ac.uk Internet Source	<1 %
24	ddd.fit.cvut.cz Internet Source	<1 %
25	journals.lww.com Internet Source	<1 %
26	Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, Yang Xiang. "A Survey of Android Malware Detection with Deep Neural Models", ACM Computing Surveys, 2021 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		