

Performance Analysis of Machine Learning Algorithms for Malware Classification

by

Raisa Hasan Bushra
18301064

Md Taukir Alam
18301277

Aniruddho Saha
18201117

Nazmus Sakib Fahim
18201166

Nabila Mourium Binty
19101082

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

Department of Computer Science and Engineering
Brac University
September 2022

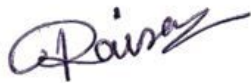
© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

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

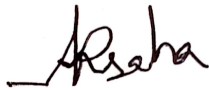
Student's Full Name & Signature:



Raisa Hasan Bushra
18301064



Md Taukir Alam
18301277



Aniruddho Saha
18201117



Nazmus Sakib Fahim
18201166



Nabila Mourium Binty
19101082

Approval


The thesis titled “Performance Analysis of Machine Learning Algorithms in Malware Classification” submitted by

1. Raisa Hasan Bushra(18301064)
2. Md Taukir Alam(18301277)
3. Aniruddho Saha(18201117)
4. Nazmus Sakib Fahim(18201166)
5. Nabila Mourium Binty(19101082)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 28, 2022.

Examining Committee:

Supervisor:
(Member)



Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Ahanaf Hassan Rodoshi
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

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

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Malware detection research has been popular over the years as the variations and complexity of malware attacks are increasing daily. Using variously Supervised and Unsupervised machine learning algorithms to detect, identify, or classify malware attacks has been proven a very effective technique for some past years. Some common and widely concerning malware attacks are Trojan, Adware, Ransomware, and Zero-day. In this paper, we used ten ML algorithms such as AdaBoost, Stochastic Gradient Descent (SGD), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), XGBoost, Logistic Regression (LR), Multi-Layer Perceptron (MLP), K-Nearest Neighbour(KNN), Support Vector Machine (SVM) for classifying software-based Trojan attacks, Ransomware, Adware and Zero-day attacks. This research was conducted on a dataset having a total sample of 12863 malware, consisting of the malware categories mentioned above, to extract features and learn patterns. Also, we showed a comparison between these ML methods and analysis based on how they classify these popular malware in this paper after testing each classifier on the selected dataset. After implementation, RF achieved the highest accuracy of 86.97%, and Gaussian NB achieved the lowest accuracy of 47.84%. MLP, XGBoost, KNN, DT, AdaBoost, SVM, LR, SGD got 83.60%, 82.59%, 80.68%, 79.63%, 73.30%, 73.22%, 67.08%, 64.40% accuracy respectively. Other than accuracy, our analysis was based on individual accuracy, precision, and F1-score, TPR, TNR, FPR, and FNR of malware classes for each ML classifier.

Keywords: Machine Learning; Malware; Trojan; Adware; Ransomware; Zero-day; Classification; Decision tree; Naive Bayes; Stochastic Gradient Descent; Random Forest; AdaBoost; XGBoost; Logistic Regression; Multi-Layer Perceptron; K-Nearest Neighbour; Support Vector Machine; Analysis

Acknowledgement

Firstly, we would like to thank Almighty for his blessing and guidance which enabled us to complete our thesis without any major problem.

Secondly, to our Supervisor Dr. Amitabha Chakrabarty sir who helped us and guided us throughout the work. With his guidance, we were able to do better in our thesis.

Thirdly, thanks to the Research Assistant, Mr. Shahriar Hossain sir for his constant guidance and encouragement during our whole work. His assistance helped us to complete our work in time.

Finally, to our parents who have been showing kind support throughout the whole journey of our University life.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
Nomenclature	x
1 Introduction	1
1.1 Thoughts behind Malware Classification	1
1.2 Research Problem	1
1.3 Aims and Objectives	3
1.4 Application area of ML algorithms	3
1.5 ML in Malware Detection and Classification	4
2 Literature Review	5
2.0.1 Malware	5
2.0.2 Related Work	6
3 Working Principles of Algorithms	9
3.1 Gaussian Naive Bayes	9
3.2 AdaBoost	9
3.3 Stochastic Gradient Descent	10
3.4 Multi Layer Perceptron	10
3.5 Decision Tree	11
3.6 Random Forest	11
3.7 K-Nearest Neighbour	12
3.8 Logistic Regression	12
3.9 Support Vector Machine	12
3.10 XGBoost	13

4	Methodology for Malware Classification	14
4.1	Input Data	15
4.2	Feature Selection	15
4.3	Data pre-processing	18
5	Experimentation	19
5.1	Algorithm Implementation	19
5.2	Performance Metrics Calculation	19
6	Result Analysis	21
6.1	Gaussian Naive Bayes	21
6.2	AdaBoost	22
6.3	Stochastic Gradient Descent	23
6.4	Multi Layer Perceptron	24
6.5	Decision Tree	24
6.6	Random Forest	25
6.7	K-Nearest Neighbour	26
6.8	Logistic Regression	27
6.9	Support Vector Machine	28
6.10	XGBoost	29
6.11	Performance Comparison	30
7	Conclusion	31
	Bibliography	35
	Appendix A Dataset	36

List of Figures

4.1	Work Flow	14
4.2	Heatmap before Feature Selection	17
4.3	Heatmap after Feature Selection	18
6.1	Validation Accuracy vs Validation Loss curve of GNB	22
6.2	Validation Accuracy vs Validation Loss curve of AdaBoost	22
6.3	Validation Accuracy curve of SGD	23
6.4	Validation Accuracy vs Validation Loss curve	24
6.5	Validation Accuracy vs Validation Loss curve of DT	25
6.6	Validation Accuracy vs Validation Loss curve of RF	26
6.7	Validation Accuracy vs Validation Loss curve of KNN	26
6.8	Validation Accuracy vs Validation Loss curve of LR	27
6.9	Validation Accuracy curve of SVM	28
6.10	Validation Accuracy vs Validation Loss curve of XGBoost	29
6.11	Comparison of Accuracy	30

List of Tables

2.1	Accuracy of ML Algorithms for Malware Detection of Previous Works	8
5.1	Confusion Matrix for Ransomware	19
6.1	Experimental Results of GNB	21
6.2	Experimental Results of AdaBoost	22
6.3	Experimental Results of SGD	23
6.4	Experimental Results of MLP	24
6.5	Experimental Results of DT	25
6.6	Experimental Results of RF	25
6.7	Experimental Results of KNN	26
6.8	Experimental Results of LR	27
6.9	Experimental Results of SVM	28
6.10	Experimental Results of XGBoost	29

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

TN True Negative

TNR True Negative Rate

TP True Positive

TPR True Positive Rate

Chapter 1

Introduction

1.1 Thoughts behind Malware Classification

Malware is an abbreviation for malicious software, is created with the intention to harm data or devices. These viruses are either a file or a code which is often supplied through a network that can infect, steal, examine or even can perform nearly any activity an attacker wishes. Malware classification is the technique of collecting all the unknown malware and grouping them based on their selective attributes or features [49]. Malware is defined as any software that performs unwanted and suspicious activities on victim computers and devices without his/her authorization. There are several types of malware variants that can steal confidential data, launch distributed denial of service (DDoS) attacks, and cause disruption to computer systems [31]. Malicious attacks surged by 700% between 2012 and 2013, according to Symantec's 2014 Annual Security Report. Additionally, in 2013, more than 552 million private identities were made public, a 493% rise from 2012. These attacks frequently included malware of some kind [9]. According to the McAfee report [4], of the fourth quarter threat, there were 75 million unique malware samples in 2011. And also, according to the Computer Economics Inc., the total global damage caused by malware is \$13.3 billion. Even though antivirus software is useful for malware protection, it is not always capable of detecting incoming malware attacks. Malware attacks can be spread through email attachments (almost 92%), android ads, different website popups, different types of torrents, or file-sharing software can also be the cause of malware attacks. The malware attacks are varied in different types. In this paper, we are focusing on four different types of malware attacks; Ransomware, Trojan attacks, Zero-Day attacks Adware. Each type of attack is similar yet can be distinguishable to some extent.

1.2 Research Problem

According to [36], malware is a very concerning threat that invades computer security and makes it vulnerable. Analyzing it is a challenge both theoretically and practically in the field of Computer Science, and there could be lots of undecidable problems to face during the research. Some problems are unpacking execution, identifying malware samples by matching against a set of known templates, and end detection of trigger-based behaviour [13]. In this work, we focused on four types of malware (Adware, Ransomware, Trojan Zero-day).

Android malware is evolving very fast, and these can escape the traditional solutions [13]. One of the ways it is spreading very fast is through mobile ads. Third-party apps are also responsible for hosting mobile malware. According to a statistic [9], mobile malware is on the rise, with the number of new mobile malware varieties increasing by 54% in 2018. Machine learning-based solutions are quite a popular tool to detect adware.

Ransomware is another deadly malware that is aimed to restrict targeted victims from accessing computer data by encrypting it with an indestructible method that can only be decoded by the attacker [47]. Even if the victim tries to delete the software, it leads to the victim losing all his/her data permanently. This paper [24] stated how traditional software's vulnerable to detecting Ransomware in executable files. In another paper, the author tried to distinguish Ransomware from other kinds of malware [23].

The trojan is another well-known malware. Trojans have been upgraded, and a variety of sophisticated tactics have been merged, making detection much more complex and time-consuming than in the past. Lack of awareness and information, as well as effective Trojan analysis techniques, have resulted in financial loss, lower productivity, and harm to organizations' reputations [1].

Zero-day is another type of threat that hasn't been encountered before. It has been created to exploit or interrupt network communications. Unsupervised algorithms, which are anomaly-based, may demonstrate low detecting performance. It has been seen that unsupervised algorithms like KNN and MLP work better than a lot of supervised algorithms. Another paper also demonstrated a previously mentioned advantage in detecting zero-day attacks using supervised algorithms; hence, a logical strategy appears to be to develop a synergy between supervised and unsupervised ML algorithms in order to design a successful model [3][41].

Therefore, with a large number of datasets that is available to us. In this paper, we are going to answer the question:

Which ML classifier has the highest accuracy in classifying certain kinds of malware?

As mentioned earlier, we are going to use ten different ML algorithms.

The research has given an answer to the above question by exploring the different malware and finding the best accurate algorithm.

1.3 Aims and Objectives

This research aimed to understand better different ML classifiers' effectiveness on different types of malware. With our large dataset, we used ten popular machine learning algorithms to evaluate the performance, along with the following objectives for this research.

1. To evaluate the use of ML Algorithms in cyber security.
2. To deeply understand the different types of malware.
3. To find the effectiveness of different ML algorithms for a vast dataset.
4. To compare the accuracy and other parameters of different ML algorithms on the classification of different malware.
5. To understand the use of Neural Network to improve the detection of malwares.

1.4 Application area of ML algorithms

Along with cyber security, ML algorithms have been proven an effective tool to solve many problems in every sector of human life. ML algorithms are vastly used in the healthcare sector. Detecting many diseases has been much easier with the help of ML classifiers. In addition to that, people use ML algorithms in other sectors, such as agriculture, transportation, and financial services. As ML algorithms work on datasets, this tool benefits data-intensive legal works also. As a result, many researchers work with ML algorithms in those sectors. Besides, many are working to bring more novel approaches to solve the same problem more effectively and efficiently.

In a recent paper [32], authors have improved the NB classifier to reduce traffic risk. They have used feature weighting and Laplace calibration for the improvement of this ML algorithm. This modified classifier works better on the large dataset than on small samples. However, this improved NB classifier has an accuracy of 92% in predicting and classifying driving risk, whereas the original NB has an accuracy of 49.5%. After that, in another research [35], an SGD-based algorithm was used to predict cancer genes. Univariate analysis was performed before applying the ML classifiers in this paper, and a log loss metric was used to evaluate the model. Other than this, Neural Network has also been used in the prediction of hereditary cancers in another paper [46]. The researchers have developed NN-based models to predict breast cancer. Furthermore, AdaBoost has been used to classify soil and, based on the soil class, recommends appropriate fertilizer to enhance the productivity of a field [11]. This algorithm identifies the richness rate and supplements of soil to prescribe the fertilizer based on the outcome. Furthermore, as the Decision Tree-based models have a low prediction accuracy level in data mining, the authors of the paper [29] aimed to improve the optimization technique of this model. They suggest a way to discretize continuous characteristics using a probabilistic approach. It vastly improves the categorization rule, eliminating incorrect classification rules to improve accuracy. Several other papers [34][20] also used DT to predict daily smoking behavior and classify the pavement's roughness. Besides, DT and LR were used to

predict the traffic congestion problem of the urban area. LR achieved 95.69%, and DT achieved 97.65% accuracy from the confusion matrix in another paper [28].

Further, these algorithms and SVM combined were also used to predict heart disease [5]. The author attempted to identify the key factors and accurately estimate the total risk by applying homogenous data mining techniques. The author also used hybrid data mining methods with the amalgamation of these algorithms to find the best result. RF was used in spam email filtering and house price prediction models [43][10]. RF utilizes the concept of ensemble learning to address complicated issues by adding multiple classifiers to boost the model's accuracy. While testing the accuracy, it was learned that the house price prediction model almost predicted the prices perfectly and only with a -/+5% price difference in some cases. The spam filtering model also performed very accurately, with an accuracy of 99.92%. Multifactorial genetic disorder diseases like cancer, dementia, and diabetes can be stopped if it is detected at early stages [45]. ML classifiers like KNN and SVM are used to predict diseases with multifactorial genetic inheritance history with an accuracy of 92.8% and 91.2%, respectively. In the paper [44], the author examines using a mathematical model as a substitute for the software thermal testing using XG Boost algorithm. XG Boost is a machine learning technique that was used to develop a mathematical model that utilizes decision trees. The author predicted the temperature of silicon heaters which are similar to IC chips on a circuit board. With the maximum error being 13.5% and a minimum error of 8.5%, the xG Boost can be seen as a valuable method for predicting IC chip temperature and reducing electronic failures caused by insufficient heat dissipation. From the above discussion, we can say that ML has a vast application area that covers almost every aspect of our life.

1.5 ML in Malware Detection and Classification

In cyber security, many researchers use a static or dynamic approach to analysis to determine whether software code is malicious or benign. Static analysis methods do not run code and instead rely on the code structure and other data features which are binary. On the contrary, dynamic analysis methods run the code to examine its execution characteristics via the network or endpoint devices [16]. In addition to that, using Machine Learning (ML) techniques enables automatic identification of malware based on their dynamic behavior and improves security [26]. In this paper, we used ten different ML algorithms. They are NB, RF, DT, MLP, Adaboost, XGBoost, KNN, LR, SVM, and SGD. After applying the algorithms, we analyzed which classifier has the best malware classification and detection efficiency. For the dataset, we merged four datasets from "[CIC-AndMal-2020] Static-Dynamic Malware Analysis" [42] containing different datasets on different malware, and we worked with the merged dataset. After merging, we applied Feature Selection to extract features and then applied ML algorithms for further analysis. The experiment result showed the effectiveness of different classifiers in classifying different malware.

Chapter 2

Literature Review

In the world of technology, malware attack is a prevalent cyber-attacks that is expanding all over the world at a threatening rate. Malware can cause many problems on a computer, mobile, or any other smart device. Some malware can occupy a great space in a device, causing a storage shortage, whereas some malware can freeze the system or crash it. Moreover, some malware can access important information and manipulate it in various ways, which dramatically threatens cyber security. For this reason, various ML algorithms are a handy tool to detect and prevent such malware. Besides, many researchers are working to ensure data security by using advanced techniques like Neural Network, Deep Learning, Data Mining, etc., which can also detect unknown malwares. As a result, using ML algorithms and other advanced techniques for detecting and classifying different malwares has become a viral work in the last many years.

2.0.1 Malware

Any file or program that is intended to cause harm to a user is classified as malware. A virus is one of the types of malware which works on its own and spreads to other programs. Another such type of malware is worms, that self replicates and spread autonomously; spyware, which discreetly collects user information and activity; a Zero-day attack is something that uses a zero-day exploit to steal or damage data from a computer or any other system; Ransomware encrypts a user's data and demands payment. Some other forms of malware are Adware, Keyloggers, Rootkits, and Backdoor viruses.

Usually, the intentions of the owners of malware are to steal personal, financial, or corporate information either to target advertisements or to have control over a particular device within a botnet.

2.0.2 Related Work

Adware

Adware is advertising-supported software that usually displays unwanted advertisements. In the research paper[30], a detection model was proposed which would protect the smart devices from adware attacks. Different data pre-processing approaches, feature selection algorithms, and ML techniques are used to detect Adware using any dataset. In this research[30], (CICAndMal2017) dataset is used, which consists of samples of benign and other malwares.

The researchers of the paper [18] used a machine learning approach based on a scheme to detect Android Adware based on static and dynamic features. Classifying each adware sample into a specific family of machine learning techniques, Adware was detected. The machine learning approach combining static and dynamic features was preferable to using static and dynamic features alone. From the Drebin dataset, APK files were obtained for adware families Hamob and Copycat.

Ransomware

According to [47], one of the problems with preventing Ransomware totally that new types of Ransomware are getting created every day, and old traditional anti-Ransomware systems are struggling against them. So the authors included Neural Network in the traditional system that can be used extensively in the creation of novel Ransomware solutions. The authors used a dataset from Github named Ransomware (Malware) Detection Using ML. The authors implemented a 10-fold cross-validation technique to generalize the model. They used accuracy, F-beta score, precision, recall and area under the ROC curve to evaluate the performance of the models. In [24], the author also stated how traditional antivirus softwares are vulnerable to detecting Ransomware in executable files. In both of these models, we can see the highest accuracy in Random Forest (RF) algorithm.

In this experiment, the author tried to distinguish Ransomware from other kinds of malwares [23]. The proposed Class Frequency and a Non-Class Frequency model is generated using the generator vector with weights. They also used six ML Algorithms RF, NB, SGD, KNN, SVM and LR, of which the detection accuracy was up to 98.65%. In this solo author paper [19], the author aimed to expose the Ransomware attacks on android phones using machine learning. This study also used the popular Machine Learning Algorithms like DT, RF, Gradient Boosting Decision Trees and also AdaBoost. The author used datasets from HelDroid and merged selected datasets to a total sample of 1923 records. Using the dataset with five attributes, the author found a high average of 98.05% accuracy rate, but the author also found decreasing in the accuracy in the Gaussian and Bernoulli on 97.6% while the Multinomial is on 81.6%.

To identify Ransomware, in paper [12], the researchers created a reverse engineering framework that incorporated feature engineering and machine learning. Their methodology is utilized for multi-level analysis to inspect in a better way and comprehend the intent of malware code parts. In this model, the author proposed two major attributes, which are Feature Generation Engine along with ML model. Also, they have used eight ML classifiers, and all of them could reach an accuracy level of more than 90%. Among them, BN obtained the highest accuracy of 97.076% and LR had the lowest accuracy of 89.183%.

Trojan

Sequential Minimal Optimization (SMO) was designed to identify Trojan Horse according to [6]. The researchers worked with datasets from VX Heaven and Offensive Computing, and they implemented Data Mining (KDD) and Knowledge Discovery to extract the patterns. While comparing the findings with other classifiers like MLP, J48, IBK, and Naive Bayes in [6], SMO got the highest TP rate which is 98.2% accuracy.

A recent work [15] shows the importance of detecting the Trojan attacks on runtime. STRIP was conducted on MNIST, CIFAR 10, GTSRB to check if it could identify Trojaned inputs. Besides, other Trojan detection works like AC, NC and SentiNet, along with STRIP, were performed to compare with each other. In this comparison, STRIP could evaluate on MNIST and CIFAR 10 datasets along with SentiNet and its computation cost and time overhead were less than that. In another research [40], Meta Neural Trojan Detection (MNTD) was introduced, which was also trained on the MNIST dataset. This model was also compared with the other algorithms used in the paper [15] and achieved the highest accuracy with less time complexity.

Further, many Supervised and Unsupervised Machine Learning algorithms are used in the detection of Trojan in hardware. According to [7], Online Learning Algorithm, which is MBW (Modified Balanced Winnow), is used for detecting hardware Trojans in real-time. In another paper [33], Random Forest Model was designed to detect hardware Trojans. However, our focus will be on software-based Trojan rather than hardware-based Trojan.

Zero-day

Zero-day attack is known as the threat of an unknown security issue. It can be either software or an application for a computer or other smart devices. The patch has not been released, or the application developers were unaware of or did not have sufficient time to address it. Zero-day is created using code obfuscation techniques that can modify the authentic code to produce multiple copies which have similar functions in different signatures.

According to [38], a methodology has been introduced to evaluate the performance of ML techniques in the detection of zero-day attacks. Zero-shot Learning (ZLS) is a new technique which is used to evaluate and improve the ML models for new or unseen data classes [39]. In this research, two NIDS datasets are used to evaluate the performance of the ML models, i.e., NF-UNSW-NB15-v2 [46] and UNSW-NB15

[35]. In work [25], an autoencoder implementation is proposed for the detection of zero-day attacks. Two well-known IDS data sets are used for evaluation—CICIDS2017 dataset and NSL-KDD. The results demonstrate a zero-day detection accuracy of 89%-99% for the NSL-KDD dataset and 75%-98% for the CICIDS2017 dataset [39]; a detailed analysis is provided on three datasets, the CIC-IDS2018, which is a better approach than other open-source data sets, which are very popular but old, KDD99 and NSL-KDD. Dataset Wednesday-21-02-2018 reached the highest accuracy of 99.999%.

<i>Reference Number</i>	<i>Algorithm</i>	<i>Ransomware</i>	<i>Adware</i>	<i>Zero-day</i>	<i>Trojan</i>
[12], [37], [7]	Naive Bayes Classifier (NB)	90.33% [12], 35% [37], 97.6% [7]	-	-	-
[8], [38], [37], [7]	Decision Tree (DT)	97.46% [38], 98% [37], 97.6% [7]	98.66% [8]	-	-
[12], [38], [37], [32], [35]	Random Forest (RF)	98.51% [12], 97.86% [38], 99% [37]	-	80.67% [32], 98.27% [35]	
[7]	Ada Boost	97.5% [7]	-	-	-
[8], [12], [38], [37]	Logistic Regression (LR)	90.27% [12], 78.54% [38], 98% [37]	83% [8]		
[8], [12], [38]	K-Nearest Neighbor (KNN)	98.06% [12], 97.46% [38]	98.05% [8]	-	-
[8], [35]	XG Boost (XGB)	-	98.10% [8]	98.513% [35]	-
[12], [35]	Support Vector Machine (SVM)	79.82% [12]	-	98.058% [35]	-
[8], [37], [32]	Multi-Layer Perceptron (MLP)	97% [37]	96.4% [8]	85.5% [32], 95.90% [32]	-
[46]	Meta Neural Trojan Detection (MNTD)	-	-	-	Average over 97% and over 90% for jumbo MNTD [46]
[14]	STRIP	-	-	-	Over 93% for several datasets [14]

Table 2.1: Accuracy of ML Algorithms for Malware Detection of Previous Works

Chapter 3

Working Principles of Algorithms

3.1 Gaussian Naive Bayes

Naive Bayes is a robust algorithm used for classification [2]. It calculates conditional class probabilities based on Bayes Theorem, which can be denoted as equation 3.1.

$$P(X|Y) = \frac{P(X).P(Y|X)}{P(Y)} \quad (3.1)$$

Here, $P(X)$ = Probability of X

$P(Y)$ = Probability of Y

$P(Y|X)$ = Probability of Y given X

$P(X|Y)$ = Probability of X given Y

Then, it computes the most probable class from a vector of training data X and sample data D. Gaussian NB approach uses Gaussian distribution X, computing its parameters from the sample data, covariance matrix, and mean vector. A particular feature is chosen from the dataset, and an assumption is made that this picked feature is strongly independent of any other features. In a supervised learning method, GNB can be very efficient and implemented in real-life situations.

3.2 AdaBoost

AdaBoost starts working by making stumps from each feature of the dataset. Here, a stump is a node with two leaves. At the initial stage, weights are calculated for each record and assigned sample weights following $w = 1/N$, where N is denoted as the number of records [48]. After that, from each stump, this algorithm will generate a decision tree and calculate its Gini and Entropy.

$$Gini = 1 - \sum_i p_i^2 \quad (3.2)$$

$$Entropy = - \sum_i p_i * \log_2 * p_i \quad (3.3)$$

In formulas 3.1 and 3.2, p_i is the probability of the corresponding class i.

Then, the first base learner will be selected from the stumps with the least Gini

and Entropy. After that, TE (Total Error) will be calculated to further calculate the performance of the stump by following the formula 3.4. TE is calculated as the summation of errors in the classified records in terms of the sample weights.

$$Performance = \frac{1}{2} \ln \left[\frac{1 - TE}{TE} \right] \quad (3.4)$$

For a correctly classified sample, the performance value will be negative, and for the wrong output, it will be positive. After calculating performance, all the weights need to be updated following formula 3.5.

$$New\ weight = Previous\ weight * e^{performance} \quad (3.5)$$

Then, normalization might be applied to bring the sum equal to 1 if it is less than 1. Thus, iterations will be run through stumps to find the lowest training error, and a prediction will be made when it is achieved.

3.3 Stochastic Gradient Descent

SGD starts from a random point, and in each iteration, it goes down its slope in steps on a function until it gains the lowest point [17]. At first, a gradient of the function is calculated. Then, a random initial point is chosen, and the gradient is updated for the parameter values. After that, the step size is calculated where $step\ size = gradient * learning\ rate$, and the parameter is updated as $new\ parameter = old\ parameter - step\ size$. Here, the learning rate is set to be a small value so that the step size does not jump down too much. Thus, iteration will take place for every point until the algorithm finds the gradient close to 0.

3.4 Multi Layer Perceptron

A multi-layer perceptron is a type of fully connected feedforward artificial neural network. For training, the dataset MLP applies the supervised backpropagation learning method. Consisting of an input layer, hidden layer, and output layer, MLP has at least three levels of nodes [11]. Each node has an activation function, and each node is connected to every other node in the layer below. An MLP is distinguished by multiple layers of input nodes connected as a directed graph between the input and output layers. The (hidden layer sizes) attribute is essential for defining the number of hidden levels and the nodes within each layer. The performance of the model is enhanced by adding layers to the hidden layers and nodes. A sigmoid activation function is used by each node in the multi-layer perception. The sigmoid activation function takes actual values as input and uses the sigmoid formula to transform them into numbers ranging from 0 and 1 [51].

$$f(x) = \frac{1}{(1 + e^{-x})} \quad (3.6)$$

Before beginning training, the network's weights are randomly assigned. The model is validated after completing the learning step with the training data. In the training

set, data are inputs (x_1, x_2), and y is the expected output of the input data with weight, w , and bias, b [21]. The output is determined by neurons and the neural network's weight. It is represented by

$$y = wx + b \tag{3.7}$$

3.5 Decision Tree

The decision tree is a supervised learning model applied in regression and classification problems. Mostly used as a classifier, it is a tree-based structure where the internal nodes denote dataset features, branches for the decision-making process, and each leaf node for the classification result [50]. A decision tree is created using an algorithmic method that finds different circumstances under which to divide a data collection. The CART algorithm (Classification and Regression Tree algorithm) is used to construct a decision tree. The tree does simple binary Yes/No questions, and based on that, it splits into subtrees. The fundamental problem that emerges while developing a decision tree is choosing the best attribute for the root node and for sub-nodes. So, a method known as attribute selection measure, or ASM, can be used to tackle these issues. By using this measurement, we can choose the ideal attribute for the tree nodes with ease.

To have the best possible outcome decision tree uses a metric measurement called entropy. Entropy is a measurement used to assess the impurity of a particular characteristic. It describes the randomness of data [50]. Using the training dataset, it can be calculated as $E(T)$ using the following equation 3.8 where P_i is distribution of each attribute [15].

$$E(T) = \sum -P_i \log P_i \tag{3.8}$$

3.6 Random Forest

Random forests are a collection of tree predictors where each tree is reliant on the values of a random vector sampled randomly and with the same distribution for all trees in the forest [8]. Each tree is trained with replacement using a single subset of the training data. It is an unsupervised algorithm that uses the bootstrap aggregation ensemble technique. Using the random forest classification model first takes input data (assuming the dataset has k sets), which takes n sets of data. Then from the given attributes, it selects some random number of attributes. Then it uses these attributes to generate output using multiple decision trees through the training method. As the name suggests, it builds a forest of decision trees from our given set of attributes, with each giving an output, and finally, it gives the result based on the majority of votes given by each tree [15]. This step is known as aggregation. Random forest is a very diverse algorithm as it does not consider all attributes/features while building a tree, as each tree is unique. Also, it reduces the risk of having dimensionality problems. We also do not need to separate the data into train and test because the decision tree generated using bootstrap would always ignore 30% of the data. Even with some limitations, it is a very robust, and

reliable model to use for maximum performance and efficiency. It is a tree-structured algorithm and can be represented as a formula 3.9 [37].

$$h(x, i_k), k = 1, 2, 3, 4, \dots, N \quad (3.9)$$

Here, N is the number of input vector x and i_k is the causative factors.

3.7 K-Nearest Neighbour

KNN is a supervised algorithm that classifies data based on proximity or distance. At first, parameter k is set to equal the number of the nearest neighbor. Then the distance is measured. Mostly, Euclidean distance is used to measure the distance between neighbors. If coordinates of two points are (x_1, y_1) and (x_2, y_2) , then it can be measured as $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Based on the distance, the K nearest neighbor is determined. Then among the K neighbors, the number of data points is counted. Finally, new data points will be assigned in the category where the neighbor is maximum. The advantage of this algorithm is its simplicity and accuracy.

3.8 Logistic Regression

Logistic Regression is used to predict the output of the categorical dependent variable. This algorithm gives output as discrete values: 0 or 1. Here a threshold value limit is set, which will round up the input values into 0s and 1s. Initially, the input values would be set to X . Then, by using Linear Model, the probabilities of the inputs would be found. Now, a threshold is to be set to 0.5. Based on the threshold value, all the values would be rounded up. The values which are above the threshold are 1, and the below values are 0. The final result would be produced as either 0 or 1 based on probabilities. The probability, P , can be computed from the following equation 3.10 where a and b are the parameters of the model and X is the independent variable.

$$P = \frac{e^{a+bX}}{1 + e^{a+bX}} \quad (3.10)$$

3.9 Support Vector Machine

SVM or Support Vector Machine is the most well-established supervised ML technique that generates a non-probabilistic classification model to determine new data categories. It is universally used for data analysis or pattern recognition. Generally, SVM is used to determine a hyperplane that separates the two classes in the training set. These classes are put into different categories. Then, the decision boundary is generated on the base of the support vectors from the two closest points. For multiple classes, the decision boundary is a hyperplane instead of a straight line. Thus, data points are classified using SVM. Although it is sometimes hard to find such a hyperplane, SVM uses kernels that include linear, non-linear, polynomial, Gaussian, and Radial Basis functions.

3.10 XGBoost

XGBoost is known as a distributed gradient boosting library that has been developed to be very effective, adaptable, and portable. It uses the Gradient Boosting framework to implement machine learning algorithms. It reduces the error rate in sequential models boosting and optimizing decision trees. Further, XG Boost operates well for discrete, unstructured, and medium-sized datasets [33]. By dealing with missing values and tree pruning, XGBoost reduces the training duration. The capacity to cross-validate data and parallel processing. The algorithm starts with the root node. Then it traverses all over the nodes using a depth-first search. The loss reduction follows as equation 3.11.

$$G = loss(F) - (loss(LB) + loss(RB)) \quad (3.11)$$

Here, G stands for gain in the branch while F represents parent node, and RB, LB denotes right node, left node respectively. This algorithm develops the efficiency of computational time.

Chapter 4

Methodology for Malware Classification

Our overall work focused on classifying different types of malware using machine learning. We used a dataset where four types of malware were available. This chapter explains the methodologies we used to complete our work. Everything was done following several steps, from merging the dataset to data pre-processing. After that, ML classifiers were implemented, and classification reports and confusion matrices were generated to calculate the performance metrics. The flowchart of our work is shown in Figure 3.1.

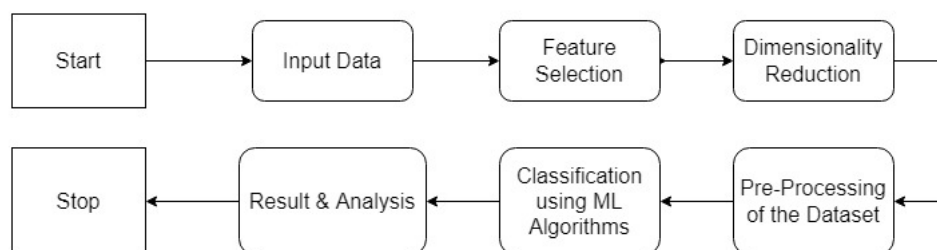


Figure 4.1: Work Flow

The classification process is to classify different malware. There are three significant steps we considered to achieve our goal. Firstly, Input data: we merged four datasets with similar labeling, categories, and characterization. Then, applied Feature Selection to reduce the dimensionality of the dataset. Data pre-processing: here, we applied several techniques like label encoding and scaling to prepare the data for applying classifiers. Then, we split the dataset into train and test sets for training classifiers on the train set and testing datasets on the test set. Prediction: In this stage, we used ten ML algorithms to classify and predict the algorithm's accuracy for this dataset.

4.1 Input Data

There are plenty of detection-based data available in the research field. However, most of the data we found detected certain malware. Our goal is to classify different types of malware, so we needed data where at least 5-6 types of malware were available in an extensive dataset. Furthermore, we had another option to merge the different malware datasets and work on the merged dataset. In that case, we also had to merge the datasets based on the features, which was tricky to achieve. We also wanted to work on a comparatively new dataset. The dataset we used [42] here meets our goals. There are 14 prominent datasets available with 191 malware families. The datasets are labeled and characterized into the corresponding family. This dataset is acquired from Androzoo dataset. This dataset is also perfect for us as all the categories and features are the same, characterized and broken down into 6. After running the malware in an emulated environment, several features are extracted. The main six categories include memory, Api, network, battery, logcat, and process. The categorized malware families are collected and divided into eight areas which also include sensitive data from media, hardware, internet, storage, etc. There are 14 different malware types of data available in this dataset; among them, we chose four based on the most available number of samples as it would give us the scope of working on a large dataset to acquire the best possible outcome of this research.

This dataset is freely available at Kaggle with proper labeling and categories. And according to our research goal, we merged the dataset to classify different malware using different ML algorithms.

4.2 Feature Selection

Feature selection denotes reducing data dimensionality to increase the algorithm's performance [22]. It is a part of data preprocessing in which unnecessary, less essential, or redundant data are removed from the dataset.

The Pearson correlation method for feature selection explains a linear relationship between two variables [27]. This method finds the relationship between class features and continuous features. High correlative variables are more linearly dependent, so similar effects on dependent variables can be observed.

A value between -1 and 1, known as a Pearson correlation, quantifies the degree to which two variables are linearly connected. Our dataset consisting of numerical values is very suitable for finding the correlation between the categories. The goal of this is to drop the features which are slightly correlated. Here the correlation values can be described as,

- When the value is 0, it means the features share no relation with each other.
- Value around 0 means the weak correlation.
- A value approaching 1 indicates a stronger positive correlation, and a value approaching -1 indicates a stronger negative correlation.

So, to understand the correlation between features, we generated a heatmap, shown in Figure 4.2. From it, we can see that some unnecessary features have 0 correlation with each other and some weakly correlated features. For generating a heatmap and implementing FS, we followed some steps, which can be written as follows,

Step 1: After importing all the essential modules and the dataset, we pass the data with `Corr()` method to generate a heatmap.

Step 2: Then we label the X/Y axis of the dataset where we have independent and dependent features.

Step 3: Then we split the data to train and test so that we will be able to overcome the overfitting issue.

Step 4: As we are working on a large dataset, we generated a 200/200 while plotting the figure of the dataset. As we generate the `Corr()` of `x_train`, our heatmap gets generated in the figure.

Step 5: It is essential to give appropriate colors to understand the correlation from looking at the picture. It gives us a better scope to visualize the relation than just looking at numbers.

Step 6: Then we used Algorithm 1 where we passed three parameters named `dataset`, `thresholdUp`, and `thresholdLow` as -0.003 to 0.003 as we wanted to drop our correlated features between these range.

Step 7: Inside the correlation matrix, we traversed through all the correlation matrix columns. Then the algorithm will compare each column with every other column and give us a value between -1 to 1 .

Step 8: From each column's correlation value, we dropped the features where the correlation value was between -0.003 to 0.003 and created a dataset of 74 columns.

Algorithm 1 Feature Drop Algorithm

```

1: Generate a correlation matrix from the dataset
2:  $i \leftarrow 1, j \leftarrow 1$ 
3:  $thresholdUP \leftarrow 0.003, thresholdLow \leftarrow -0.003$ 
4: while  $i \leq len(matrix.columns)$  do
5:   while  $j \leq i$  do
6:     if  $matrix[i][j] \geq thresholdLow$  and  $matrix[i][j] \leq thresholdUp$  then
7:       delete  $matrix.columns[i]$  from the dataset
8:     end if
9:   end while
10: end while

```

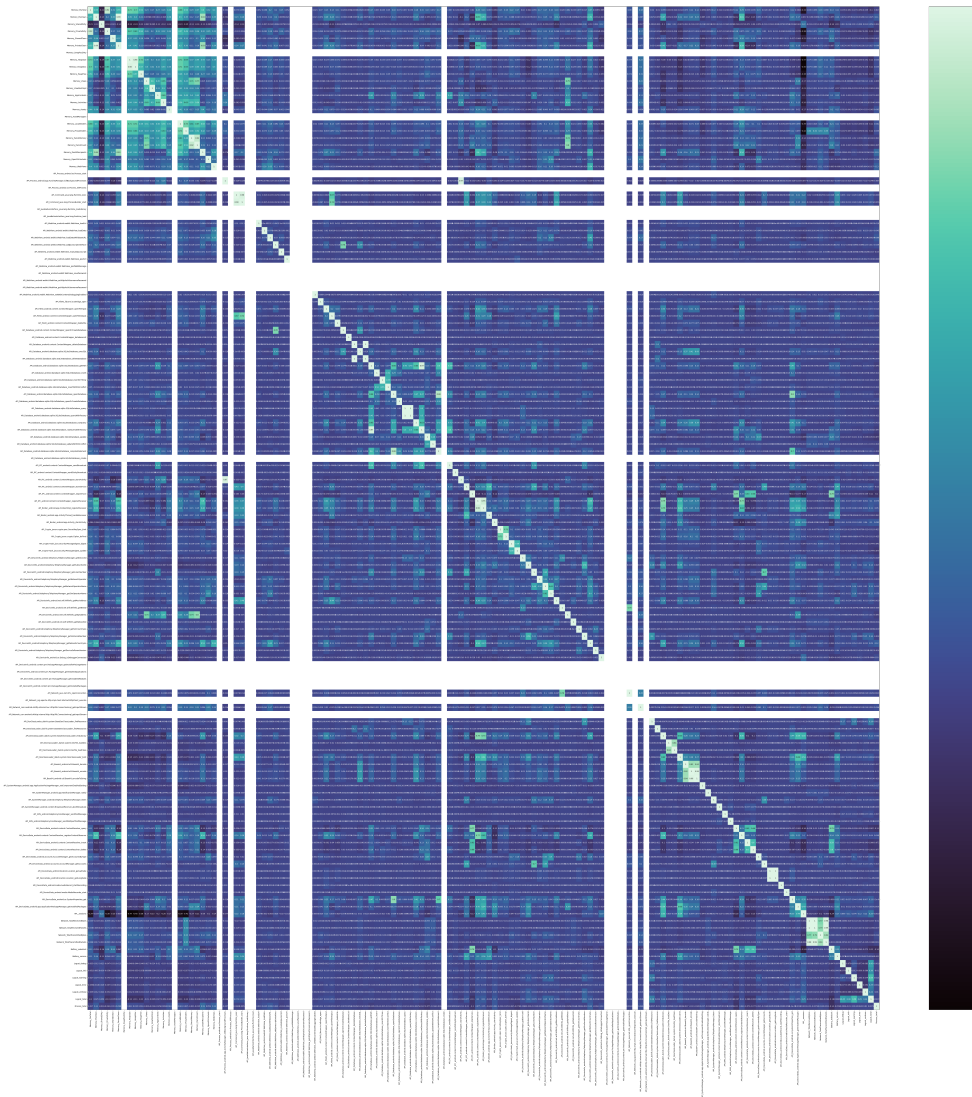


Figure 4.2: Heatmap before Feature Selection

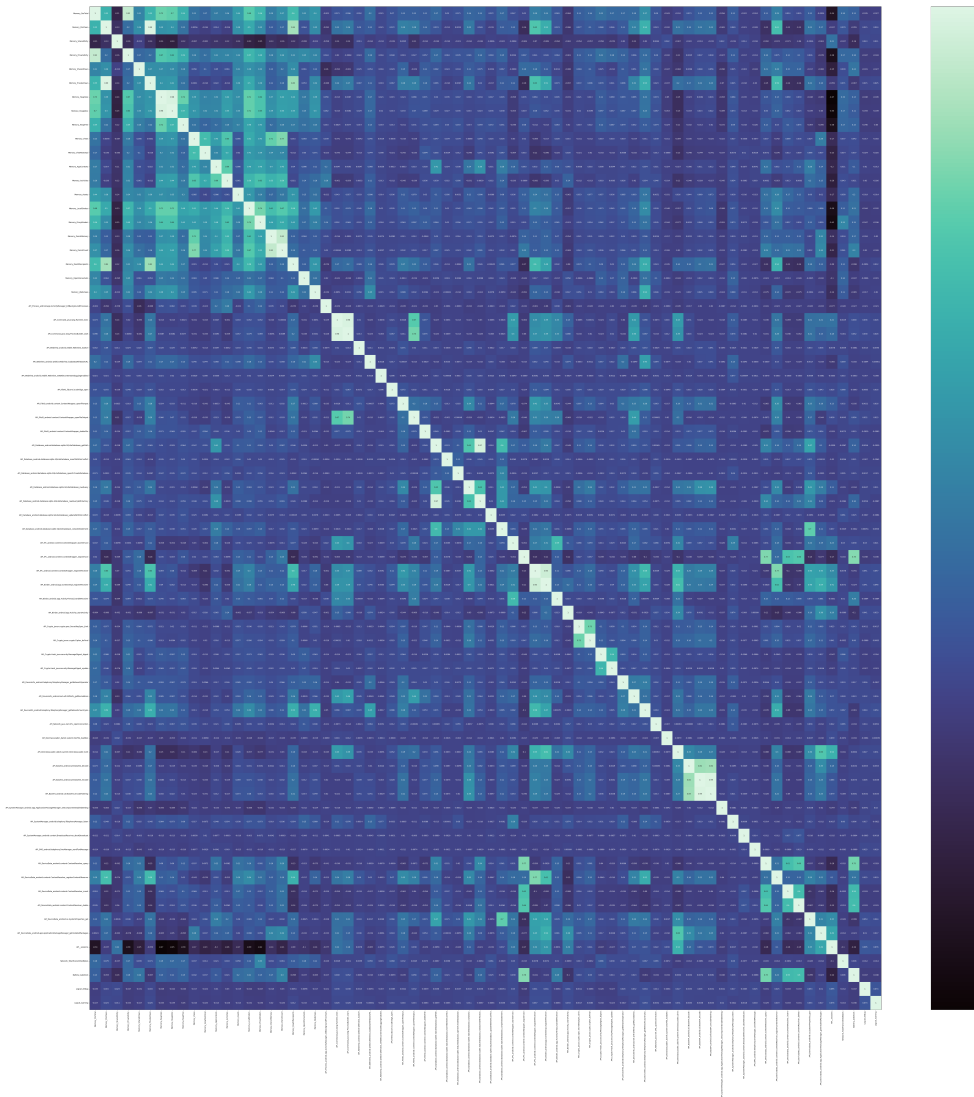


Figure 4.3: Heatmap after Feature Selection

4.3 Data pre-processing

After Feature Selection, we proceeded to pre-process our dataset to make it convenient for classifiers. Firstly, we found the categorical feature of our dataset and encoded it using the Label Encoding technique. Then, we separated this feature from our dataset and applied MinMax scaler to rescale the range of the features between 0 and 1. It helps the classifiers to learn efficiently and understand the problem in a better way. After scaling, we used train-test-split to split the dataset by an 80% - 20% ratio so that we could fit the classifiers on the train sets and make predictions on the test sets.

Chapter 5

Experimentation

5.1 Algorithm Implementation

We used python libraries and packages like Numpy, Pandas, Matplotlib, and Seaborn for our coding. After pre-processing the data, we applied ten classifiers one by one to generate classification reports and confusion matrices and test the accuracy of the algorithms. For that, we imported classifiers from the Sklearn library and created an object for each classifier. Then the objects were fitted on the train sets, and predictions were made on the test set. By comparing the test prediction and actual value, accuracy was achieved.

5.2 Performance Metrics Calculation

We generated a Confusion Matrix (CM) for each classifier to identify each class's TP, TN, FP, and FN. The dimension of each CM is 4*4 as we have four classes. Here, for a particular class in a confusion matrix, TP is the true positive value, TN is the summation of true negative values, FP is the summation of false positive values, and FN is the summation of all the false negative values.

		<i>Prediction</i>			
		Ransomware	Zero-day	Trojan	Adware
<i>Parameter</i>	Ransomware	TP	FN	FN	FN
	Zero-day	FP	TN	TN	TN
	Trojan	FP	TN	TN	TN
	Adware	FP	TN	TN	TN
<i>Actual</i>					

Table 5.1: Confusion Matrix for Ransomware

In Table 5.1, the determination of parameters for Ransomware CM is shown. TP is the value where actual and predicted values intersect with each other. FN is the summation of values of that corresponding row, excluding the TP value. Similarly,

FP is the summation of the corresponding column values without the TP value. Furthermore, TN is the summation of all the other values of the CM, excluding the corresponding row and column values. Thus, we calculated TP, FP, TN, and FN for each class from CM for every ML classifier. Lastly, We calculated TPR, TNR, FNR, FPR, and accuracy by following the formulas.

$$TPR = \frac{TP}{TP + FN} \quad (5.1)$$

$$FNR = 1 - TPR \quad (5.2)$$

$$TNR = \frac{TN}{TN + FP} \quad (5.3)$$

$$FPR = 1 - TNR \quad (5.4)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.5)$$

Chapter 6

Result Analysis

In this chapter, we showed the performance metrics of each classifier with the parameters of accuracy, recall, precision, F1-score, TPR, TNR, FPR, and FNR, along with the analysis. Apart from it, we also generated a validation accuracy curve and validation loss curve to evaluate each model.

6.1 Gaussian Naive Bayes

<i>GaussianNB</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.7248	0.84	0.38	0.53	0.9521	0.0479	0.6167	0.3833
<i>Trojan</i>	0.6867	0.26	0.93	0.41	0.6555	0.3445	0.0741	0.9259
<i>Ransomware</i>	0.7349	0.57	0.63	0.60	0.7824	0.2176	0.3694	0.6306
<i>Zero-day</i>	0.8103	0.36	0.12	0.18	0.9535	0.0465	0.8761	0.1239

Table 6.1: Experimental Results of GNB

In Table 6.1, it can be seen from the accuracy column that GNB could not perform well in classification. GNB achieved a good precision in classifying Adware, whereas others' precision was very low compared to it. Precision and recall are essential parameters for classifying malware to understand its effectiveness. Like precision, GNB's recall values (TPR) were also low for most cases; exceptions could be seen for Trojan. Additionally, F-1 scores of GNB classification were low as it depends on precision and recall values. On the other hand, the miss rates (FNR) were higher for Adware and Zero-day, meaning these malware could not be classified well by GNB. Additionally, GNB underfitted the dataset in both the validation accuracy and validation loss curves in Figure 6.1, this algorithm performed poorly in malware classification based on the performance metrics, and its overall accuracy was 47.84%.

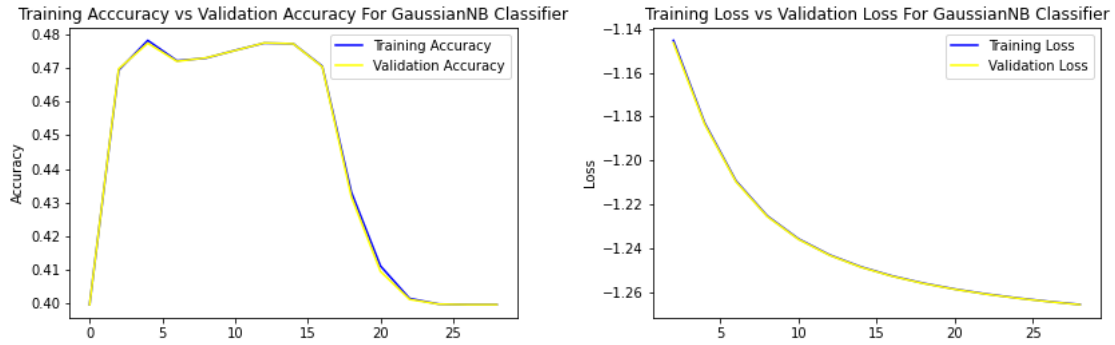


Figure 6.1: Validation Accuracy vs Validation Loss curve of GNB

6.2 AdaBoost

<i>AdaBoost</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8247	0.75	0.85	0.79	0.8091	0.1909	0.1518	0.8482
<i>Trojan</i>	0.9728	0.85	0.93	0.89	0.978	0.022	0.0673	0.9327
<i>Ransomware</i>	0.8321	0.72	0.76	0.74	0.8655	0.1345	0.2413	0.7587
<i>Zero-day</i>	0.8364	0.55	0.29	0.38	0.9512	0.0488	0.714	0.286

Table 6.2: Experimental Results of AdaBoost

Table 6.2 shows that the accuracy column performed pretty well in classification. Analyzing the precision column, we get the idea of how many predicted cases are positive. We see the precision of Trojan is comparatively higher than the others. Similarly, Adaboost's recall value is lower except for Zero-day. F1 score depends on precision and recall values. So, its value is low compared to precision and recall. On the contrary, the miss rate (FNR) is high for zero-day, which means zero-day is not classified well by Adaboost. Besides, with AdaBoost, there is no overfitting or underfitting issue, which can be seen in Figure 6.2. Finally, we see that this algorithm has an accuracy of 73.30%.

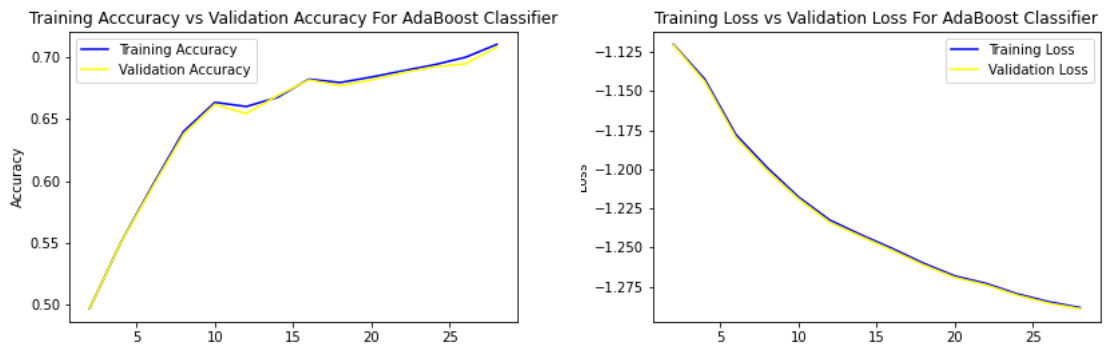


Figure 6.2: Validation Accuracy vs Validation Loss curve of AdaBoost

6.3 Stochastic Gradient Descent

<i>SGD</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.6988	0.57	0.93	0.71	0.8091	0.4539	0.0653	0.9347
<i>Trojan</i>	0.9273	0.72	0.72	0.72	0.978	0.0419	0.2818	0.7182
<i>Ransomware</i>	0.8127	0.82	0.52	0.64	0.8655	0.0535	0.4761	0.5239
<i>Zero-day</i>	0.8492	0.72	0.12	0.2.	0.9512	0.0088	0.8849	0.1151

Table 6.3: Experimental Results of SGD

Table 6.3 shows that the accuracy column performed better in classifying malware, except for Adware. The precision column gives us the ratio of correctly classified positive samples to the total number of classified positive samples. The precision of Ransomware was comparatively higher. The recall measures the model's ability to detect positive samples, and here Adware's recall value was higher than others. As the F1 score was the weighted average of precision and recall, its value was lower than precision and recall. However, the miss rate (FNR) was higher in Zero-day and Ransomware, and from its overall performance, we can say that it could not classify the dataset satisfactorily. Eventually, we get an accuracy of 64.40% from SGD. In Figure 6.3, there was no underfitting, but there was a slightly overfitting issue. In SGD, a validation loss curve can not be generated because the vanishing gradient problem causes NAN values in validation loss calculation.

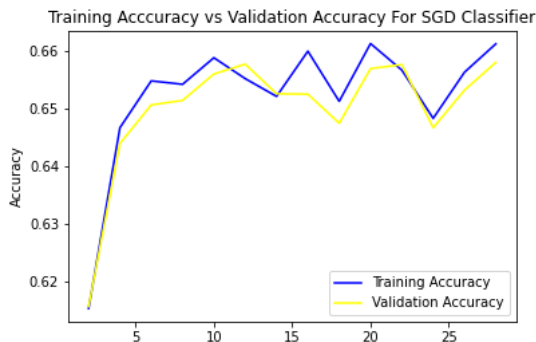


Figure 6.3: Validation Accuracy curve of SGD

6.4 Multi Layer Perceptron

<i>MLP</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8997	0.87	0.88	0.88	0.912	0.088	0.1187	0.8813
<i>Trojan</i>	0.9782	0.89	0.93	0.91	0.9851	0.0149	0.0741	0.9259
<i>Ransomware</i>	0.9192	0.89	0.85	0.87	0.952	0.048	0.153	0.847
<i>Zero-day</i>	0.8749	0.63	0.65	0.64	0.9216	0.0784	0.3491	0.6509

Table 6.4: Experimental Results of MLP

Observing Table 6.4, we can say that Trojan and Ransomware’s accuracies are pretty good. The accuracies of the other two malware were high also. MLP classified Adware, Trojan, and Ransomware with good Precision and Recall(TPR), except for Zero-day. The same was for F1-score, since its value relies on precision and recall values. The miss rates(FNR) for all four malware were low, indicating these classes were well classified using MLP with an accuracy of 83.60%. Also, in the case of the MLP validation accuracy curve, both the training and testing accuracy lines in Fig 6.4 are closely aligned. Thus the overfitting issue, in this case, can be negligible. However, MLP was noticed to be slightly overfitted in the validation loss curve.

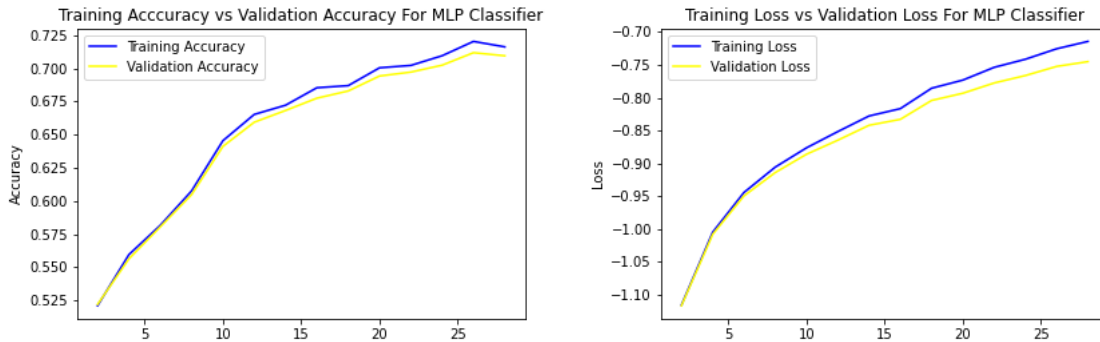


Figure 6.4: Validation Accuracy vs Validation Loss curve

6.5 Decision Tree

The accuracy column of Table 6.5 shows that all the malware achieved high accuracy. However, precision and recall for Zero-day were low but high for all the other three malware. Similarly, F1-score for that three malware were also high but low for Zero-day. Miss rates(FNR) for all four malware were lower, letting us know that DT well classified the malware with an accuracy of 79.63%. From Figure 6.5, we can see that, in the case of DT, the training accuracy was higher for both the validation accuracy and validation loss curves, indicating overfitting.

<i>DecisionTree</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8686	0.86	0.81	0.83	0.91	0.09	0.1936	0.8064
<i>Trojan</i>	0.9759	0.87	0.93	0.90	0.9824	0.0176	0.0741	0.9259
<i>Ransomware</i>	0.8865	0.80	0.84	0.82	0.9062	0.0938	0.1567	0.8433
<i>Zero-day</i>	0.8616	0.60	0.60	0.60	0.9159	0.0841	0.3986	0.6014

Table 6.5: Experimental Results of DT

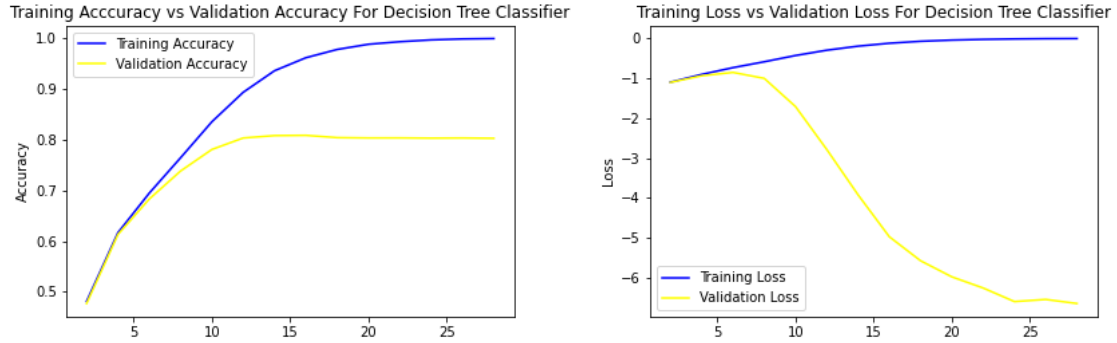


Figure 6.5: Validation Accuracy vs Validation Loss curve of DT

6.6 Random Forest

<i>RandomForest</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.9129	0.85	0.96	0.90	0.8835	0.1165	0.0428	0.9572
<i>Trojan</i>	0.9864	0.93	0.95	0.94	0.9908	0.0092	0.0471	0.9529
<i>Ransomware</i>	0.9355	0.90	0.89	0.90	0.9559	0.0441	0.1095	0.8905
<i>Zero-day</i>	0.9048	0.82	0.57	0.68	0.9737	0.0263	0.4257	0.5743

Table 6.6: Experimental Results of RF

Table 6.6 shows that Adware, Trojan, Ransomware, and Zero-day accuracy were all above 90%. Other performance matrices, such as Precision, Recall, F1-score, were high for Adware, Trojan, and Ransomware but low in the case of Zero-day. Miss rates(FNR) were also low for all the malware. Thus we can say these malware are well classified using the RF classifier with an accuracy of 86.98%, which is the highest among all the other ML classifiers. Since the training accuracy in Figure 6.6 was higher in both curves, the RF is overfitting our dataset.

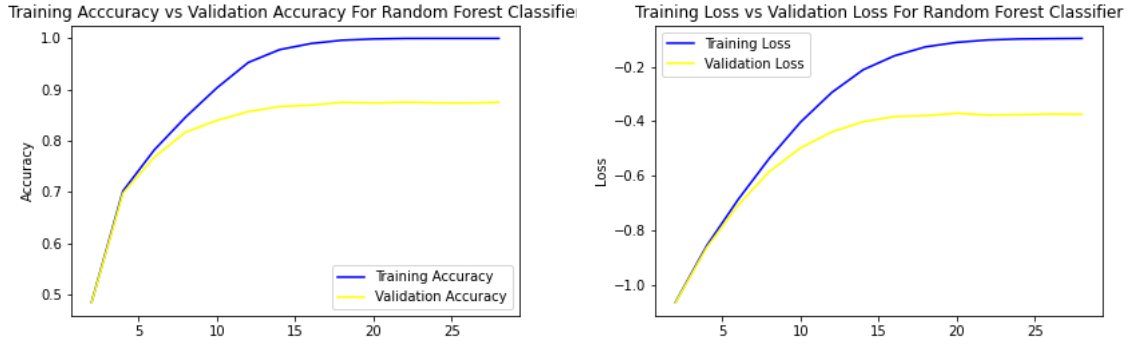


Figure 6.6: Validation Accuracy vs Validation Loss curve of RF

6.7 K-Nearest Neighbour

<i>KNeighbors</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8772	0.83	0.87	0.85	0.8841	0.1159	0.1333	0.8667
<i>Trojan</i>	0.967	0.83	0.90	0.86	0.9754	0.0246	0.0976	0.9024
<i>Ransomware</i>	0.8947	0.82	0.85	0.83	0.9158	0.0842	0.1517	0.8483
<i>Zero-day</i>	0.8749	0.68	0.53	0.59	0.9469	0.0531	0.4707	0.5293

Table 6.7: Experimental Results of KNN

Accuracy for all the malware were high, according to Table 6.7. Other performance metrics like Precision, Recall, and F1-score were high for other malware except for Zero-day. FNR or miss rates were low for all the malware, so it is safe to say that KNN classified these malware well enough with an accuracy of 80.68%. Finally, according to Figure 6.7, we can conclude that the KNN classifier is overfitting our dataset since both curves' training accuracy and loss were higher than validation accuracy and loss.

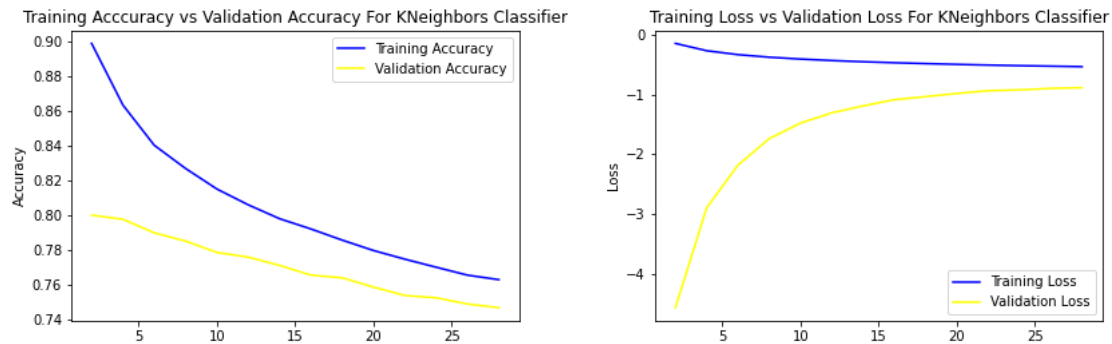


Figure 6.7: Validation Accuracy vs Validation Loss curve of KNN

6.8 Logistic Regression

<i>LogisticRegression</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.7742	0.67	0.86	0.75	0.7184	0.2816	0.142	0.858
<i>Trojan</i>	0.9398	0.71	0.82	0.76	0.9552	0.0448	0.1785	0.8215
<i>Ransomware</i>	0.7995	0.67	0.70	0.69	0.8451	0.1549	0.301	0.699
<i>Zero-day</i>	0.8282	0.51	0.09	0.15	0.9831	0.0169	0.9144	0.0856

Table 6.8: Experimental Results of LR

The LR did not perform very well for this dataset while classifying malware, as it only had 67.08% accuracy. According to Table 6.8, the accuracy of Adware, Trojan, Ransomware, and Zero-day are all above 90%. Other performance matrices, such as Precision, Recall, and F1-score, are high for Adware, Trojan, and Ransomware but low in the case of Zero-day. Miss rates(FNR) are also low for all the malware. Thus we can say, these malware are well classified using RF classifier. It had excellent accuracy in Trojan. Precision and recall had similar results in three malware, but it performed very poorly in Zero-day. We had similar results in the F1-score, like the precision and recall. The FNR in Zero-day is much higher, indicating that Logistic regression did not classify the malware correctly. Also, in the LR validation accuracy curve, the training and testing accuracy lines shown in Figure 6.8 are closely aligned. The validation vs. training loss curve also showed that this algorithm did not face any overfitting or underfitting issues, similar to the validation accuracy curves.

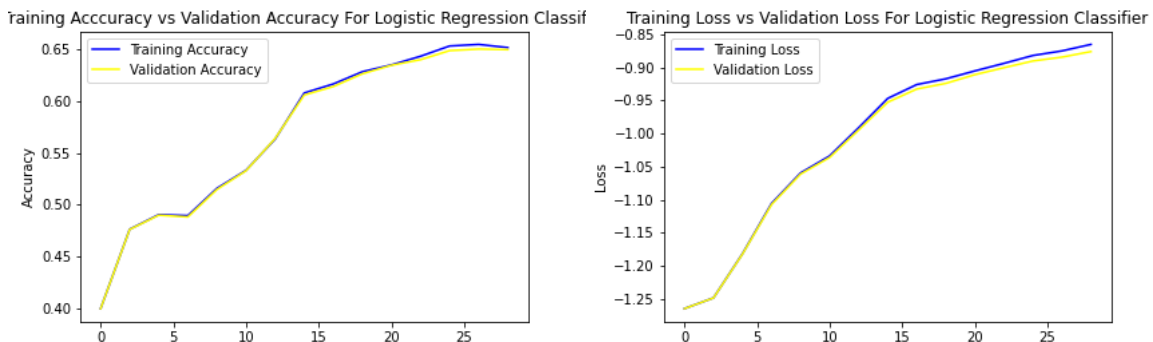


Figure 6.8: Validation Accuracy vs Validation Loss curve of LR

6.9 Support Vector Machine

<i>SVM</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8212	0.73	0.88	0.80	0.7832	0.2168	0.1216	0.8784
<i>Trojan</i>	0.9518	0.74	0.91	0.81	0.9578	0.0422	0.0943	0.9075
<i>Ransomware</i>	0.8492	0.75	0.77	0.76	0.8841	0.1159	0.2276	0.7724
<i>Zero-day</i>	0.8422	0.63	0.20	0.31	0.9751	0.0249	0.795	0.205

Table 6.9: Experimental Results of SVM

In Table 6.9, we got average results in all classifiers using SVM, with Trojan having the highest accuracy. Overall, this algorithm achieved 73.2% accuracy. However, it failed to hold out the precision and recall in the zero-day result. It got much worse in Zero-day, with values dropping way below the other three. The FNR values were also much lower for all algorithms apart from Zero-day, which indicates that the algorithm performed very poorly for Zero-day. TNR values are also relatively high, demonstrating that this algorithm could correctly categorize the false values in most cases. Unlike most other algorithms we used in this research, we could not generate a validation loss curve for SVM as the training loss and testing loss values became NAN. However, this algorithm did not face any overfitting or underfitting issues in the validation accuracy curve.

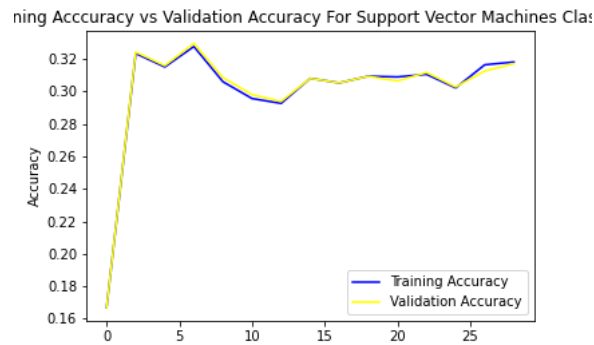


Figure 6.9: Validation Accuracy curve of SVM

6.10 XGBoost

<i>XGBoost</i>								
	Accuracy	Precision	Recall	F1-score	TNR	FPR	FNR	TPR
<i>Adware</i>	0.8931	0.83	0.92	0.87	0.8751	0.1249	0.0798	0.9202
<i>Trojan</i>	0.9771	0.87	0.95	0.90	0.9811	0.0189	0.0539	0.9461
<i>Ransomware</i>	0.9024	0.82	0.88	0.85	0.9135	0.0865	0.1219	0.8781
<i>Zero-day</i>	0.8791	0.76	0.43	0.55	0.9723	0.0277	0.5676	0.4324

Table 6.10: Experimental Results of XGBoost

XGBoost achieved 82.59% accuracy for the classification. It has a noticeably good individual accuracy, and the precision was over 80% on average for all four classes. Apart from these two parameters, we can see in Table 6.10 that the miss rate (FNR) is very low for Adware, Ransomware, and Trojan, which indicates that this algorithm performed pretty well. Additionally, TNR values are also very high, indicating that this algorithm could mostly classify the false values in a correct way. However, XGBoost faced an overfitting issue which can be seen in Figure 6.10. From the validation accuracy curve, we can see that training accuracy was more than the testing accuracy. Similarly, from the loss validation curve, the testing loss was more than the training loss, making the algorithm overfitting the dataset.

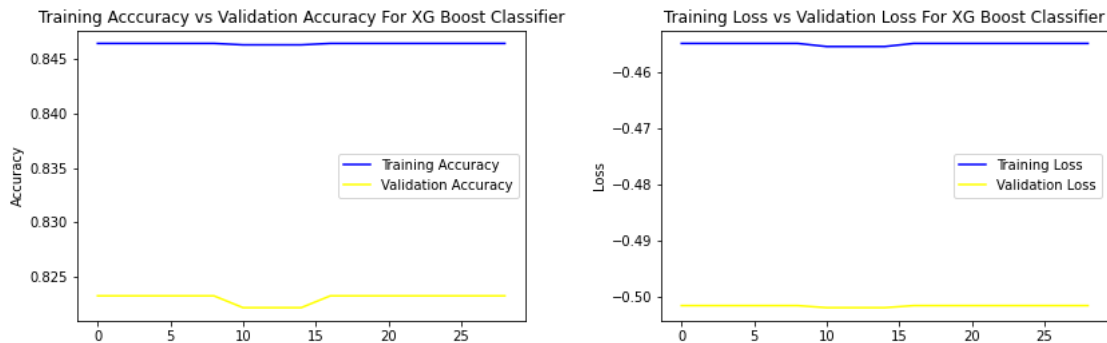


Figure 6.10: Validation Accuracy vs Validation Loss curve of XGBoost

6.11 Performance Comparison

Analyzing the performance of the algorithms, we can see that our three best fitting algorithms are RF, MLP, and XGBoost, with an accuracy of 86.98%, 83.60%, and 82.59%, respectively. Based on the performance matrices, RF performed best in all categories of malware, with an accuracy of over 90% for each class. Although RF requires more computations and takes longer to train the model because of its multi-tree structure, it obtained the highest accuracy as it takes the majority vote in the decision tree while calculating the accuracy. As we found in some previous research that unsupervised algorithms perform better than supervised ones, we have found quite a similar case. The Neural Network based multilayer architecture named the Multilayer Perceptron algorithm performed way better than most supervised models. Because of hidden layers and nodes and increased accuracy in each iteration, it provides better results. Like RF, MLP also obtained good accuracy in all four categories along with good Precision and Recall(TPR). With noticeably good individual accuracy, and the precision was over 80% on average for all four classes XGBoost is the third best performing algorithm in our experiment.

Average performing algorithms like K-NN, Decision Tree, Adaboost, and Support Vector Machine have gained 80.68%,79.63%,73.30%, and 73.20% accuracy, respectively. DT is a tree-based algorithm, but it does not take the majority voting like RF, thus having lesser accuracy. Even though AdaBoost is in the same family as XGBoost, it takes more computation time than XGBoost as it takes some irrelevant data into account while performing. KNN and SVM performed quite similarly, as shown in the table. Both obtained more than 90% accuracy in one category and an average of 80% in the other three categories. LR, SGD, and Gaussian Naive Bayes are the worst performing algorithm in our with an accuracy of 67.08%,64.40%, and 47.84%, respectively. From the experiment, it can be stated that these algorithms (RF, MLP, XGBoost) performed well in classifying the malware.

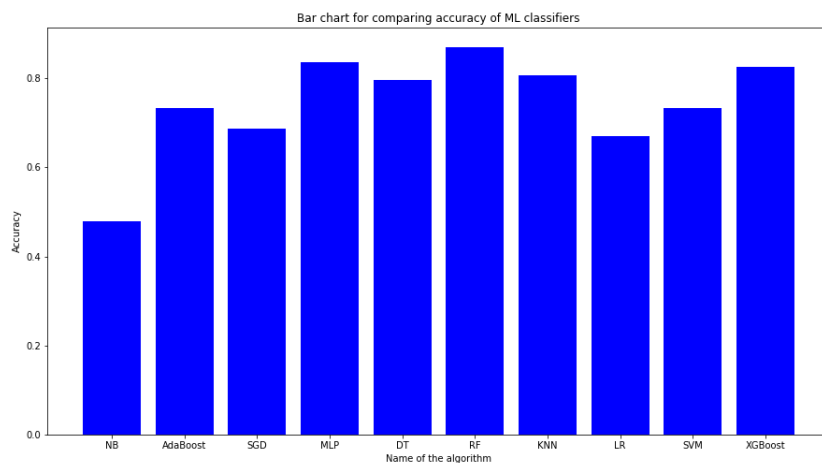


Figure 6.11: Comparison of Accuracy

Chapter 7

Conclusion

With an increased number of malware attacks on an everyday basis around the world, a new malware classification system is needed to protect our information from falling into the wrong hands. Lots of malware share some aspects, but it is essential to classify them to their respective families. Our research focused on using different tools and techniques to identify the best model we can use to classify their variants. Thus, our research used ten different algorithms: NB, AdaBoost, SGD, MLP, DT, RF, KNN, LR, SVM, and XGBoost.

The experiment showed that our proposed method could classify malware with up to 86.98% accuracy using RF. In addition, its accuracy and precision for each class were also very high compared to other algorithms. So, RF performed the best for our dataset. Figure 6.11 shows the accuracy comparison of ten algorithms. From this experiment, we can distinguish which algorithm can be used to find the possible outcome of classifying malware. The dataset we used, which is comparatively new, can be used for future reference for malware classification. Our comparison model can give fellow researchers an idea of which algorithms are best suited for classifying malware.

Our contributions can be summarized as follows:

- We have worked on a comparatively new dataset containing different malware. We have pre-processed this dataset to find the best possible result.
- We have used ten different algorithms to classify different malware.
- We compared them to find the best possible algorithm to classify malware.
- We have also shown validation accuracy and validation loss curves to determine if the algorithms face any overfitting or underfitting issues.
- The highest accuracy of 86.7%, and the average malware classification was more than 70% while considering the combined dataset.

In the future, we want to develop a model based on RF to detect and classify different types of malware more efficiently in real-time.

Bibliography

- [1] S. Mitropoulos, D. Patsos, and C. Douligeris, “On incident handling and response: A state-of-the-art approach,” *Computers & Security*, vol. 25, no. 5, pp. 351–370, 2006.
- [2] R. M. Moraes and L. S. Machado, “Gaussian naive bayes for online training assessment in virtual reality-based simulators,” *Mathware & Soft Computing*, vol. 16, no. 2, pp. 123–132, 2009.
- [3] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, “Combining supervised and unsupervised learning for zero-day malware detection,” in *2013 Proceedings IEEE INFOCOM*, IEEE, 2013, pp. 2022–2030.
- [4] C. Lim and L. Lukito, “Malware attacks intelligence in higher education networks,” *ISICO 2013*, vol. 2013, 2013.
- [5] T. Mythili, D. Mukherji, N. Padalia, and A. Naidu, “A heart disease prediction model using svm-decision trees-logistic regression (sdl),” *International Journal of Computer Applications*, vol. 68, no. 16, 2013.
- [6] M. Mohd Saudi, A. M. Abuzaid, B. M. Taib, and Z. H. Abdullah, “Designing a new model for trojan horse detection using sequential minimal optimization,” in *Advanced Computer and Communication Engineering Technology*, Springer, 2015, pp. 739–746.
- [7] A. Kulkarni, Y. Pino, and T. Mohsenin, “Adaptive real-time trojan detection framework through machine learning,” in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, 2016, pp. 120–123.
- [8] K. Prabha and S. S. Sree, “A survey on ips methods and techniques,” *International Journal of Computer Science Issues (IJCSI)*, vol. 13, no. 2, p. 38, 2016.
- [9] S. Pai, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, “Clustering for malware classification,” *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 2, pp. 95–107, 2017.
- [10] E. Dada and S. Joseph, “Random forests machine learning technique for email spam filtering,” *Semin Ser*, vol. 9, no. 1, pp. 29–36, 2018.
- [11] P. K. GT, J. Sabeena, *et al.*, “Agriculture soil classification and fertilizer recommendation using adaboost and bagging approaches,” in *2018 IADS International Conference on Computing, Communications & Data Engineering (CCODE)*, 2018, pp. 7–8.
- [12] S. Poudyal, K. P. Subedi, and D. Dasgupta, “A framework for analyzing ransomware using machine learning,” in *2018 IEEE symposium series on computational intelligence (SSCI)*, IEEE, 2018, pp. 1692–1699.

- [13] A. A. Selçuk, F. Orhan, and B. Batur, “Intractable problems in malware analysis and practical solutions,” 2018.
- [14] D. C. Dobhal, P. Das, and K. Aswal, “Detection of android adwares by using machine learning algorithms,” *Detection of Android Adwares by using Machine Learning Algorithms*, vol. 8, no. 4S, pp. 17–21, Apr. 2019. DOI: 10.35940/ijeat.d1005.0484s19. [Online]. Available: <https://www.ijeat.org/wp-content/uploads/papers/v8i4s/D10050484S19.pdf> (visited on 09/21/2022).
- [15] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, “Strip: A defence against trojan attacks on deep neural networks,” in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 113–125.
- [16] S. Saad, W. Briguglio, and H. Elmiligi, “The curious case of machine learning in malware detection,” *arXiv preprint arXiv:1905.07573*, 2019.
- [17] A. V. Srinivasan, *Stochastic gradient descent - clearly explained*!! Sep. 2019. [Online]. Available: <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31> (visited on 09/21/2022).
- [18] S. Suresh, F. Di Troia, K. Potika, and M. Stamp, “An analysis of android adware,” *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 3, pp. 147–160, 2019.
- [19] O. B. Victoriano, “Exposing android ransomware using machine learning,” in *Proceedings of the 2019 International Conference on Information System and System Management*, 2019, pp. 32–37.
- [20] Y. Zhang, J. Liu, Z. Zhang, and J. Huang, “Prediction of daily smoking behavior based on decision tree machine learning algorithm,” in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, IEEE, 2019, pp. 330–333.
- [21] S. Abirami and P. Chitra, *Multilayer perceptron*, 2020. [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron> (visited on 09/21/2022).
- [22] S. Asim, A. Shah, H. Shabbir, S. u. Rehman, and M. Waqas, “A comparative study of feature selection approaches: 2016-2020,” *International Journal of Scientific and Engineering Research*, vol. 11, p. 469, Feb. 2020.
- [23] S. I. Bae, G. B. Lee, and E. G. Im, “Ransomware detection using machine learning algorithms,” *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, e5422, 2020.
- [24] V. G. Ganta, G. V. Harish, V. P. Kumar, and G. R. K. Rao, “Ransomware detection in executable files using machine learning,” in *2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, IEEE, 2020, pp. 282–286.
- [25] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, “Utilising deep learning techniques for effective zero-day attack detection,” *Electronics*, vol. 9, no. 10, p. 1684, 2020.
- [26] F. Noorbehbahani and M. Saberi, “Ransomware detection with semi-supervised learning,” in *2020 10th International Conference on Computer and Knowledge Engineering (ICCCKE)*, IEEE, 2020, pp. 024–029.

- [27] F. Z. Okwonu, B. L. Asaju, and F. I. Arunaye, “Breakdown analysis of pearson correlation coefficient and robust correlation methods,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 917, 2020, p. 012065.
- [28] T. S. Tamir, G. Xiong, Z. Li, H. Tao, Z. Shen, B. Hu, and H. M. Menkir, “Traffic congestion prediction using decision tree, logistic regression and neural networks,” *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 512–517, 2020.
- [29] S. Zhifang and L. Yi, “Optimization of decision tree machine learning strategy in data analysis,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1693, 2020, p. 012219.
- [30] O. S. A. Aboosh and O. A. I. Aldabbagh, “Android adware detection model based on machine learning techniques,” in *2021 International Conference on Computing and Communications Applications and Technologies (I3CAT)*, IEEE, 2021, pp. 98–104.
- [31] Ö. Aslan and A. A. Yilmaz, “A new malware classification framework based on deep learning algorithms,” *Ieee Access*, vol. 9, pp. 87936–87951, 2021.
- [32] H. Chen, S. Hu, R. Hua, and X. Zhao, “Improved naive bayes classification algorithm for traffic risk management,” *EURASIP Journal on Advances in Signal Processing*, vol. 2021, no. 1, pp. 1–12, 2021.
- [33] N. S. Chockaiah, S. Kayal, J. K. Malar, P. Kirithika, and M. N. Devi, “Hardware trojan detection using machine learning technique,” in *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, Springer, 2021, pp. 415–423.
- [34] H. Han, T. Zhang, Q. Dong, X. Chen, and Y. Wang, “Pavement roughness level classification based on logistic and decision tree machine learnings,” in *Green and Intelligent Technologies for Sustainable and Smart Asphalt Pavements*, CRC Press, 2021, pp. 400–405.
- [35] A. Kandula, “R, s., & s, n., performing uni-variate analysis on cancer gene mutation data using sgd optimized logistic regression,” *International Journal of Engineering Trends and Technology*, vol. 69, no. 2, pp. 59–67, 2021.
- [36] M. Naseer, J. F. Rusdi, N. M. Shanono, S. Salam, Z. B. Muslim, N. A. Abu, and I. Abadi, “Malware detection: Issues and challenges,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1807, 2021, p. 012011.
- [37] Q. B. Pham, S. Chandra Pal, R. Chakraborty, A. Saha, S. Janizadeh, K. Ahmadi, K. M. Khedher, D. T. Anh, J. P. Tiefenbacher, and A. Bannari, “Predicting landslide susceptibility based on decision tree machine learning models under climate and land use changes,” *Geocarto International*, pp. 1–27, 2021.
- [38] M. Sarhan, S. Layeghy, M. Gallagher, and M. Portmann, “From zero-shot machine learning to zero-day attack detection,” *arXiv preprint arXiv:2109.14868*, 2021.
- [39] B. M. Serinelli, A. Collen, and N. A. Nijdam, “On the analysis of open source datasets: Validating ids implementation for well-known and zero day attack detection,” *Procedia Computer Science*, vol. 191, pp. 192–199, 2021.

- [40] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, “Detecting ai trojans using meta neural analysis,” in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2021, pp. 103–120.
- [41] T. Zoppi and A. Ceccarelli, “Prepare for trouble and make it double! supervised–unsupervised stacking for anomaly-based intrusion detection,” *Journal of Network and Computer Applications*, vol. 189, p. 103 106, 2021.
- [42] A. Zorzetto, [*cic-andmal-2020*] *static-dynamic malware analysis*, Dec. 2021. [Online]. Available: <https://www.kaggle.com/datasets/albertozorzetto/cic-andmal-2020-dynamic-static-analysis> (visited on 09/21/2022).
- [43] A. B. Adetunji, O. N. Akande, F. A. Ajala, O. Oyewo, Y. F. Akande, and G. Oluwadara, “House price prediction using random forest machine learning technique,” *Procedia Computer Science*, vol. 199, pp. 806–813, 2022, ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2022.01.100>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050922001016> (visited on 09/21/2022).
- [44] S. Durgam, A. Bhosale, V. Bhosale, R. Deshpande, P. Sutar, and S. Kamble, “Effective computational approach for optimization of temperature on printed circuit board,” *Journal of The Institution of Engineers (India): Series C*, pp. 1–14, 2022.
- [45] T. M. Ghazal, H. Al Hamadi, M. Umar Nasir, M. Gollapalli, M. Zubair, M. Adnan Khan, C. Yeob Yeun, *et al.*, “Supervised machine learning empowered multifactorial genetic inheritance disorder prediction,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022.
- [46] Z. Guan, G. Parmigiani, D. Braun, and L. Trippa, “Prediction of hereditary cancers using neural networks,” *The Annals of Applied Statistics*, vol. 16, no. 1, pp. 495–520, 2022.
- [47] M. Masum, M. J. H. Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, “Ransomware classification and detection with machine learning algorithms,” in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2022, pp. 0316–0322.
- [48] G. L. Team, *The ultimate guide to adaboost algorithm: What is adaboost algorithm?* Jan. 2022. [Online]. Available: https://www.mygreatlearning.com/blog/adaboost-algorithm/?fbclid=IwAR0713_7oz6yBuELeIvidPgnT6S7uJPK%201X5mwRy9rXVOc6Enad4pKdp3rqQ# (visited on 09/21/2022).
- [49] F. Zhong, Z. Chen, M. Xu, G. Zhang, D. Yu, and X. Cheng, “Malware-on-the-brain: Illuminating malware byte codes with images for malware classification,” *IEEE Transactions on Computers*, 2022.
- [50] *Machine learning decision tree classification algorithm - javatpoint*. [Online]. Available: <https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm> (visited on 09/21/2022).
- [51] *Multi-layer perceptron in tensorflow - javatpoint*. [Online]. Available: <https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow> (visited on 09/21/2022).

Dataset

- [1] The dataset consisting of four types of malware is publicly available at <https://drive.google.com/file/d/1MnFy2M0IXuMAOmruPSGiRNfEt8O09vjC/view?usp=sharing>.