Performance Analysis of Machine Learning Classifiers for Detecting PE Malware

by

ABM.Adnan Azmee
16101155
Pranto Protim Choudhury
16101062
Md.Aosaful Alam
16101061
Orko Dutta
16101022

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 and Engineering

Department of Computer Science and Engineering
Brac University
December 2019

Declaration

It is hereby declared that

1. The thesis submitted is 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:

ABM. Adnan Azmee 16101155	Md. Aosaful Alam 16101061
Pranto Protim Choudhury 16101062	Orko Dutta 16101022

Approval

The thesis titled "Performance Analysis of Machine learning Classifiers for detecting PE malware" submitted by

1. ABM. Adnan Azmee (16101155)
2. Md. Aosaful Alam (16101061)
3. Pranto Protim Choudhury (16101062)
4. Orko Dutta (16101022)

Of Fall, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December 10, 2019.

Examining Committee:

Supervisor:
(Member)

DR.Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

DR.Md.Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

DR.Mahbubul Alam Majumdar
Professor
Department of Computer Science and Engineering
BRAC University

Abstract

In this modern era of technology, securing and protecting one's data has been a major concern and needs to be focused on. Malware is a program that is designed to cause harm and malware analysis is one of the paramount focused points under the sight of cyber forensic professionals and network administrations. The degree of the harm brought about by malignant programming varies to a great extent. If this happens at home to a random person then that may lead to some loss of irrelevant or unimportant information but for a corporate network, it can lead to loss of valuable business data. The existing research does focus on some few machine learning algorithms to detect malware and very few of them worked with Portable Executables (PE) files. However, we worked on the PE files and also for real-time computation, a client-server model was developed by using Flask to detect malware or benign. In this paper, we mainly focused on top classification algorithms and compare their accuracy to find out which one is giving the best result according to the dataset and also compare among these algorithms. Top machine learning classification algorithms were used alongside neural networks such as Artificial Neural Network, XGBoost, Support Vector Machine, Extra Tree Classifier, etc. The experimental result shows that XGBoost achieved the highest accuracy of 98.62 percent when compared with other approaches. Thus, to provide a better solution for this kind of anomalies, we have been interested in researching malware detection and want to contribute to building strong and protective cybersecurity.

Keywords: Malware Detection; Machine learning; Data protection; XGBoost; Support Vector Machine; Extra Tree Classifier; Artificial Neural Network; Flask ; Client-Server Model.

Dedication

We would like to dedicate our thesis report to our parents for their constant support. Special gratitude towards our close friends.

Acknowledgement

We are highly indebted to Muhammad Iqbal Hossain for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project. Our thanks and appreciations also go to people who have willingly helped us out with their abilities in this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Malware or malicious software, a dangerous computer code, intended to disturb, cripple or take control of the computer system without the approval of the user. The term is utilized to portray a lot of dangers on the web and it comes in numerous structures generally covered up in another record or camouflaged as a harmless application. It takes advantage of the technical faults or vulnerabilities in the operating system, hardware, and software. Malware is frequently used to take assets from the PC or attempting to take some significant data, records or cash from an individual.

Most of the time certain individuals make malware for their interests and advantages and then it spread quickly through the internet. Malware has been there on the internet for quite a long amount of time and "Brain" is considered to be the first virus in the history on the personal computer (pc). It was originated by certain Pakistani young boys in their late teens for their motivations and purposes. Amjad Farooq Alvi was the mind behind the coding section of Brain[1]. Brain influences the IBM PC by supplanting the boot sector of a floppy disk with a duplicate of the infection. Elk Cloner, another famous malware founded in 1982 and often referred to as the first computer virus, was found on MAC. Elk Cloner was a boot sector virus that spread via floppy discs. The infection was connected to a game and when played the 50th time the infection would enact making the screen go dark and showing a poem on the screen.

Every malware is not disastrous but it can cause certain limits of distress like it can cause the laptop or computer to slow down or run at a slow pace and can also cause irritating conduct like creating a series of pop-up advertisements. Most of the time it redirects to different websites and starts playing unwanted videos or songs.The virus, Jerusalem, was originated back in 1987 and in the MS-DOS operating systems, it started the files to infect and washout various programs on the 13th of any Friday of any month [2]. CIH was released in 1998, a dangerous virus designed to cause havoc. It activated on the anniversary of the Chernobyl nuclear accident. It has the potential to overwrite the hard drive which will lead to loss of important documents[3].These were few of the oldest known viruses that infected computers and cause huge damage back then. As time passes, viruses improved and become more difficult to control and stop spreading.

In this 21st century, both malware and its spreading rate are growing rapidly along with the advancement of technology. Some of the most disturbing viruses of this century are Koobface Virus, Cabir Virus, Anna Kournikova virus, Cryptolocker, Backoff, Cerber and many more. Most of them were created for fun proposes but later it started to spread rapidly without control and some of them were created on purpose to gain information or other necessary data from others. Malware and viruses are also used in cyber warfare between countries. Iran and Saudi Arabia have been in cyber warfare for more than 10 years. In recent years, against Saudi Arabia, Iran has organized disparaging computer viruses which will cause damage to its oil industry. Therefore, malware and virus can also be used in deadly cases to destroy a community or even a country.

However, every malware is not disastrous but it can cause certain limits of distress like it can cause the laptop or computer to slow down or run at a slow pace and can also cause irritating conduct like creating a series of pop-up advertisements. Most of the time it redirects to different websites and starts playing unwanted videos or songs.

1.1 Motivation

Much research has been conducted on making new techniques and strategies to assemble, study and ease noxious code as malware is spreading at an alarming rate on the web. Beyond a shadow of a doubt, it is basic to accumulate and think about malware found on the Internet. Be that as it may, it is much more essential to create moderation and location strategies in light of the bits of knowledge picked up from the investigation work. Unfortunately, current host-based detection methods undergo from weak detection models. These models focus on the highlights of a particular malware occasion and are regularly effectively evadable by obfuscation or polymorphism. Additionally, finders that check for the nearness of a grouping of framework calls displayed by a malware example are frequently evadable by framework call reordering. To address the inadequacies of weak models, a few powerful discovery approaches have been suggested that expect to recognize the conduct shown by a malware family. Albeit promising, these methodologies are tragically too hindered to even think about being utilized as constant finders on the end host, and they regularly require lumbering virtual machine innovation. To overcome this major issue, we along with our thesis supervisor highly believed that we could use several machine learning algorithms to detect malware efficiently and more accurately. To study a huge dataset consisting of enormous volumes of data machine learning algorithms is very effective and time-saving and it also finds straightforward models and examples that humans will find difficult to get. We are intending to use as many machine learning algorithms as possible and also find the accuracy rate of them and create confusion matrix to compare the results with one another. This has supported us exceptionally and we accept that it is conceivable to identify malware effectively and diminish the harms that it can cause to the distinctive individual or a general public all in all.

1.2 Problem Statement

Malware is a program designed solely to cause problems and as days pass more malware is created in addition to the old existing ones. Antivirus is often unaware of any new virus or malware that is being spread through the internet and by the time a solution comes, users have already been affected by the new virus. And this can lead to the loss of useful information and money. Saving personal data is everyone's concern, but in this era of technology, it seems quite impossible. This malicious malware is the reason for losing billions of dollars as well as data. Cybercriminals will take an expected thirty three billion records by 2023 as indicated by a recent report from Juniper Exploration [4]. Every year billions of data are being stolen from various sites and some of them are used even for bad purposes. Almost 60 million Americans have been influenced by data fraud as per a 2018 online overview by The Harris Survey. A similar overview demonstrates about fifteen million purchasers experienced data fraud in 2017 [5].

The number of cyber-attacks is increasing daily with an exponential rate. According to the Symantec Internet Security Threat report 2019, the use of destructive malware by different groups increased by about twenty- percent in 2018[6]. Cisecurity report says that from December 2018 to January 2019 the number of malware activities increased by sixty-one percent[7]. Four thousand ransomware attacks occur daily according to the FBI report[8]. The attacks are causing panic among users, according to a Gallup study, more than seventy percent of Americans are worried about losing personal or financial data by getting hacked[9]. These attacks are also causing great financial loss. In the Accenture report, they say that a company on average costs 2.4 million us dollar due to malware attacks[10].

To detect, remove and prevent malware different antivirus are used. There are various approaches followed by the antivirus to detect malware. Most of the antivirus use signature-based detection system. During the scan, an antivirus checks the signature of the file and tries to match the pattern of it with the malware signature database if a match is found the file is declared as malware. But malware whose signature is not present in the database can infect the system. Many antivirus companies claim that their database is being updated periodically but it cannot prevent a "zero-day" attack. The signature-based antivirus fails to detect malware in such cases. The effectiveness of antivirus is decreasing day by day because attackers are finding a new way to evade the antivirus. When the antivirus company finds out about a new malware they need to reverse engineer it and extract the signature which is then added in their database, but by the time it's added in the database a lot of users are already under attack. Antivirus vendors are struggling to keep up with attacks. As we see that the traditional signature-based anti viruses are not efficient; we propose a malware prediction system based on machine learning. Our system uses the ability of machine learning algorithms to detect and predict different types of malware.

1.3 Objectives and Contributions

In our thesis, we carry out a comparative study on different machine learning classification algorithms in detecting malware and benign PE (Portable Executable) files. We used a dataset from kaggle which was built using python library (PE files) and contains malicious and benign data of PE files. At first from the given dataset we perform data pre-processing and then implement them on the machine learning algorithms we select to work with. In our research, we used several machine learning algorithms to assemble several classifiers and then we used those classifiers to detect the PE malware and after that, we compared the performance of classifiers in accurately identifying malware. We compared different machine algorithms and tried to figure out which algorithm has the highest accuracy to detect the malware in real-time execution by implementing a client-server model.

Malware discovery is urgent with malware's commonness on the internet since it works as an early notice framework for the PC secure in regards to malware and digital assaults. It keeps programmers out of the PC and keeps the data from getting traded off. Both organizations and common users are suffering to get protected from the malware in the cyber-space, which emphasizes the importance of developing well-organized methods of malware detection. Therefore, our approaches were to build a malware detection solution that will consequently distinguish and assemble unidentified files on a range of malware weightiness and to build better detection results based on higher accuracy. We have implemented a client-server model using flask as the tool. Here, a client will upload a file and the server will analyze the file using the JSON type data structure and detect the class and time of the provided test data. In this way, the client will be aware of the malicious file and take necessary precautions regarding it. This may even lead to saving the client's personal information from getting robbed.

1.4 Thesis orientation

Chapter 1 - Introduction where motivation, problem statement, objectives and contribution are discussed.

Chapter 2 - Related works where background and literature review are discussed. In the literature review part, we reviewed the previous work on malware detection.

Chapter 3 - Our Proposed Approach, here we discussed our methodology to detect the malware, flask server setup, and real-time implementation.

Chapter 4 - Includes the Dataset description where we discussed our dataset in details.

Chapter 5 - Algorithms where we discussed the algorithms used in our work.

Chapter 6 - Includes Result Analysis and Data Visualization, where we visualized our data and analyzed the result obtained from our algorithms.

Chapter 7 - Conclusion and Future works where the conclusion consists of our work till now and future works include the scope of improvement.

Chapter 2

Related Work

We divided related work into background and literature review.

2.1 Background

The term malware, however, was not introduced until the mid-eighties. Fred Cohen, considered as the father of “computer virus” used this term in 1986. 33 percent of all the computers in the world have been infected by malware according to the statistics of the Anti-phishing workgroup[11]. A few articles additionally expressed that misfortunes because of cybercrime including malware are foreseen to hit six trillion USD every year by 2021. Early malware was crude, regularly spreading altogether disconnected through floppy circles conveyed from PC to PC by humans. As systems administration and the web developed, virus creators rushed to adjust their malicious code and exploit the new communication medium. Creeper, Wabbit, Morris Worm, etc are some of the early versions of malware. Somewhere in the range of 2000 to2010, malware developed fundamentally, both in number and in how quickly diseases spread. Toward the beginning of the new thousand years, the Internet and worms were standing out as truly newsworthy over the globe. Afterward, we saw a historic increment in malware toolboxes, including the notorious Sony rootkit, which was contributory in malware creators incorporating rootkits in most present-day malware [11]. Crimeware units pointed explicitly at sites likewise rose in fame, and the quantity of traded off sites heightened correspondingly. SQL injection assaults turned into a machine learning danger, asserting well known unfortunate casualties, for example, IKEA. Some vital malware released in the gap between 2000 to 2010 were ILOVEYOU Worm, SQL Slammer Worm, Cabir Virus, etc. Among 2010 and right now, we’ve watched critical advancement in the complexity of malware. Composed wrongdoing and state support increased the game significantly with huge, well-subsidized improvement groups. Some of the well-known malware is described very little below.

Table 2.1: Well Known Malwares

	MALWARE	DESCRIPTION
1	ADWARE	This is software that displays unwanted advertising on a computer or mobile device in the form of pop-up advertisements or they may redirect one's browser to certain websites. Most of the time it does not cause any direct harm to user's device but can be annoying for the user.
2	BROWSER HIJACKER	Hijacking a browser means that malicious software has redirected the computer's browser to a different website usually generates to visit a certain website or lead to a malicious website that will download malware on the computer.
3	SPYWARE	This malware is designed for spying purposes as the name suggests. It hides on the computers or laptops and monitors everything the user do and it also looks after web activities, access email and can even steal the username and passwords for different platforms.
4	RANSOMWARE	With ransomware, someone can lock up another person's computer holding it hostage and forcing that person to pay a lot of money to get the files back. And this is one of the reasons that a user should always regularly backup files on the computer. Ransomware harm costs will climb to $11.5 billion in 2019 and businesses will succumb to a ransomware assault like clockwork around then [1].
5	WORMS	The main objective of worms is to spread as many copies of itself in any way possible from computer to computer. A worm can replicate itself without any human interaction and it does not need to attach itself to a program to cause damage. Worms can modify and delete files and even inject additional malware onto the computer.
6	BANKING TROJAN	A banking Trojan is malware that hides on the computer and gets in the middle of the conversation the user is having and can steal important information such as bank password and can also make you think that you are taking to your bank and get you to hand over your personal information even your PIN. In the end, it steals your money.
7	BACKDOOR TROJAN	This creates a backdoor on the user's computer allowing the attacker to access the machine to get control of it. It usually uploads stolen data and even downloads more malware on the computer.
8	DOWNLOADER TROJAN	This type of malware can download any type of information the attacker wants and can also inject more malware to your computer. This can lead to permanent loss of valuable information from the user's computer.
9	INFO STEALER TROJAN	This type of malware has just one purpose and that is to steal data from the infected computer. Once a PC has been infected by this malware can easily steal most of the information from the computer.
10	REMOTE ACCESS CONTROL	This is a special type of malware mainly designed to give the attacker full control over the victim's computer, that is, once a computer is infected the attacker now has full control and can do anything such as sending emails, steal all type of passwords, etc.
11	DDoS ATTACK	DDoS stands for distributed denial of service and is designed to take down an entire network by flooding it with traffic.
12	MACRO VIRUSES	These are the type of viruses that are written specifically to alter macros which are a common command that word processing program use. Once opened macros can

Most of the malware is created to steal sensitive and private information and data from a victim often to make money illegally. Malware is used mainly for illegal purposes such as cyber espionage, cyber warfare, cyber vandalism, hacking, and many other reasons. Malware can make section onto the client's PC employing flawed document downloads, visiting contaminated sites or through an email containing a considerate connection or attachment. A typical kind of malware is viruses. The virus frequently starts on the web and regularly spread when downloading a file contaminated with a virus distributed file-sharing or email connection.

Moreover, above mentioned well-subsidized improvement groups keep on developing today, creating progressed malware with avoidance strategies that defeat numerous regular enemies or malware frameworks. Penetrating production lines and military frameworks turned into a typical reality, and the adaptation of malware developed quickly with emotional development in ransomware and other illicit plans. Some advanced malware is Stuxnet Worm, Cryptolocker, Cerber, Ransomware, etc. These malware attacks create a huge loss and take control of personal data. SQL SLAMMER has caused serious havoc in the United States of America (USA) in 2003. It is thought that it cost those who were hit in all about one billion dollars. It affected web servers leading to bank ATMs crashing and messed with continental airlines electronic ticketing and it blocked twenty-seven million people from the internet. It also infected seventy-five thousand servers in ten minutes[12].Sircam made Microsoft windows system its target. It spread far and wide as it sent itself to people in the user's email address book. This virus would choose random files from the user computer and send them to others. This was particularly bad for the governments who do not want random files being sent out. It affected twelve percent of computers in North America and eleven percent of computers in Europe. CNET writes that it costs one billion dollars in damages related to cleaning infected systems and lost productivity[13]. Confiker is said to be the most destructive worms of all and it cost those infected around nine billion dollars. It was discovered in 2008 and has in its lifetime infected as many as fifteen million computers. It affected police departments in the UK as well as the military and messed with their operations. It infects the windows operating system and it lasted a long time as it kept getting past patches and antivirus software[14].

In 1999 "Melissa" malware targeted Microsoft users by sharing a file and told the users to share it with fifty more users. Because of this virus financial cost was eighty million USD. ILOVEYOU Worm affected 45 million computers in two days and the financial cost was ten billion[15]. Another recent malware is "Ransomware". There were 181.5 million attacks in the first six months of 2018[16].

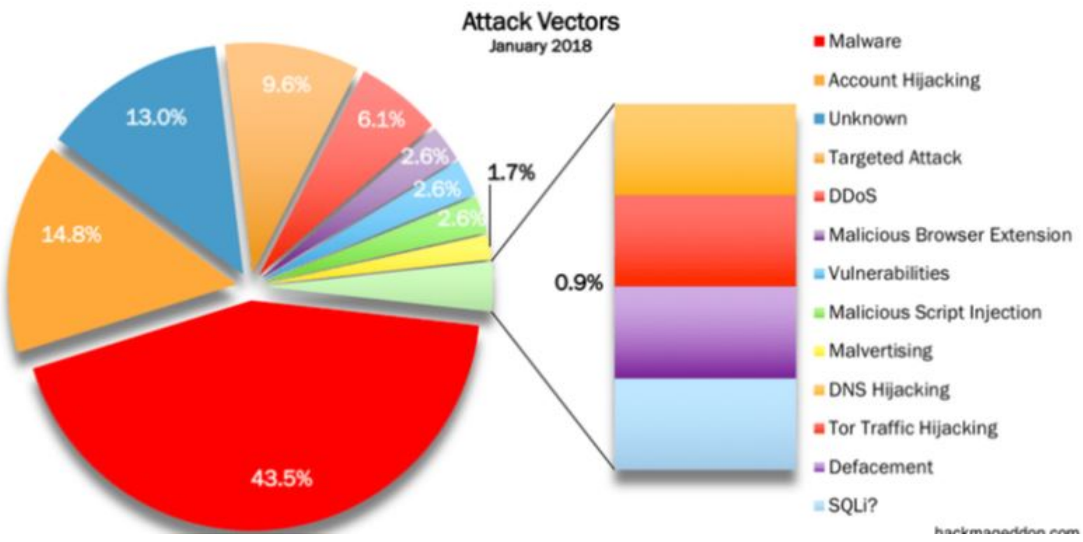


Figure 2.1: Cyber Attacks 2018 Statistics[17]

To prevent the malware attacks lots of anti-viruses are here but these antivirus soft wares are not enough to prevent malware attacks as the malware is evolving day by day. The software can become outdated if the software is not getting updated frequently. As a result, unknown malware will not be recognized by the software. That's where machine learning emerged. Machine learning looks at the dataset to find out both malicious and favorable code and attempts to understand which files are good and which are contains malware. The machine learning included attempts to settle on choices about whether dissected code is unsafe dependent on a progression of attributes. A few characteristics may rank higher than different qualities. So code that is resolved to be considerate may have a few attributes that the product considers to be a potential sign of malware. Malware is developing quickly, so the calculations must advance quickly also. It's a consistent, continuous procedure.

There has been researched and arrangement of machine learning malware examination for a long time now. We catch wind of machine learning significantly more often times nowadays on the grounds that compelling innovation is a lot less expensive at this point. Machine learning anti-malware programming can't be customer-driven, on the grounds that a customer PC or cell phone is presented to a lot littler, progressively restricted examples of malware. Legitimate machine learning needs Big Data preparing and cloud-based frameworks. Cloud servers are a lot less expensive and progressively accessible presently, so machine learning malware examination is more open than any other time in recent memory[18]. Nowadays, lots of malware detection process is happening through machine learning. There are many approaches and the evolution of machine learning algorithm is increasing day by day as the neural network field is also emerging. Deep learning/Neural networks add more revolutionary impact on the malware detection process.

2.2 Literature Review

In paper[19], they worked on portable executables and tried to figure out which of the files are malware and created an integrated feature set based on raw and derived inputs. Moreover, they had used only six algorithms to compare between integrated and raw feature set. Furthermore, they used the 10fold cross-validation technique like us. They created two datasets with the help of virus-share, Windows XP and Windows 7. One of them had two thousand seven hundred twenty-two malware and two thousand four hundred eighty-eight benign data and another one had one hundred twenty-nine malware data and thirty benign data. Lastly, they changed feature numbers several times and always found that their feature set always gave more desirable results than actual feature set. In our paper, we use more algorithms than them, and our data set is more ethical because we collected it from kaggle. And, we have more data than their data set.

In paper[20], they made a model which was intended to help in the network security of enterprises. They did it because they felt that the signature-based antivirus system could be enough for household purposes but might be a threatening issue for networks of enterprises. However, they used only three algorithms to compare with their model and worked on only five thousand data. Their model gave ninety-eight percent accuracy but only worked well for enterprise networks, not with the personal computers of home. Also, they did not mention any processing techniques like PCA, LDA but we used some of those techniques in our work. We are also ahead of them in comparison to the highest accuracy rate of algorithms and the number of data in the data set.

A model which detects pe malware is created by authors of paper[21].Their system worked in real-time. Actually, they worked in three steps, these are feature extraction, selection, and classification. For feature extraction,pefile, which is a python module had been used. But they only used five hundred fifty-two data which is way less than us. Moreover, they used the chi-square test, which found the relation between features with statistics and eventually works well with the small amount of data. If there are lots of data, it can lead to erroneous conclusions whereas our model did well with a large amount of data. We use a variety of algorithms like decision trees, boosting, neural networking. Lastly, their accuracy rate was 97.25percent, we found more accuracy in our work even with a large amount of data.

Another real-time malware detection work like us was done on paper[22]. They extracted only thirty-five features, again less than our number of features. They used only four algorithms; cnn,mlp,svm and random forest and netmate technique for feature extraction. It is another paper using fewer algorithms and got less accuracy rate than us. Each of their algorithms gave an accuracy rate of more than eighty-five percent. Whereas in our work, we got a more than ninety percent accuracy rate in several algorithms. And they did not use any feature reduction technique.

In paper[23], research had been done on machine learning malware detection. In the early stage of our work, we got different ideas from this paper. According to them, the heuristic machine learning is better than signature method which can not stop new malware and dynamic method which has time and space complexity They made the comparison on several steps which are necessary for machine learning malware detection. Firstly, they compare among different feature extraction techniques like signature-based, dll function call, binary sequence, assembly sequence, pe file header, machine activity matrix, entropy signal. Secondly, they showed their survey on feature selection methods like information gain, redundant feature removal, principal component analysis, random forest, self-organizing feature map, wavelet transform. Lastly, they discussed three algorithms, these are supported machine vector, random forest, and artificial neural network and they compare accuracy rate based on feature selection and algorithms.

A machine learning technique which is hybrid in nature used for the purpose of malware detection in paper[24]. They had the target of tracing and categorizing malware and had a better accuracy rate. For gaining that, they collected 5.05 GB benign along with 1.28 GB malware data and extract of n-gram and pe type of data. They only used three supervised classifiers (J48, Random Forest, Naive Bayes) and one unsupervised classifier(Self-organizing feature map). They used the information gain technique for feature selection and came to know that random forest gave more accuracy rate than all classifiers and that is of ninety-four percent(less than us).

And in paper[25], they created a big data set and opened it for all for research purposes. Their data set contained three hundred thousand malicious data, three hundred thousand benign data, and three hundred thousand data which were not labeled. For test purposes, they reduced it to one hundred thousand malicious data and three hundred thousand benign data and make another data set. However, they used eight types of features, only used LightGBM and proved that they gave better performance than another deep learning model, malcov.

Chapter 3

Our Proposed Approach

3.1 Machine Learning Methodology

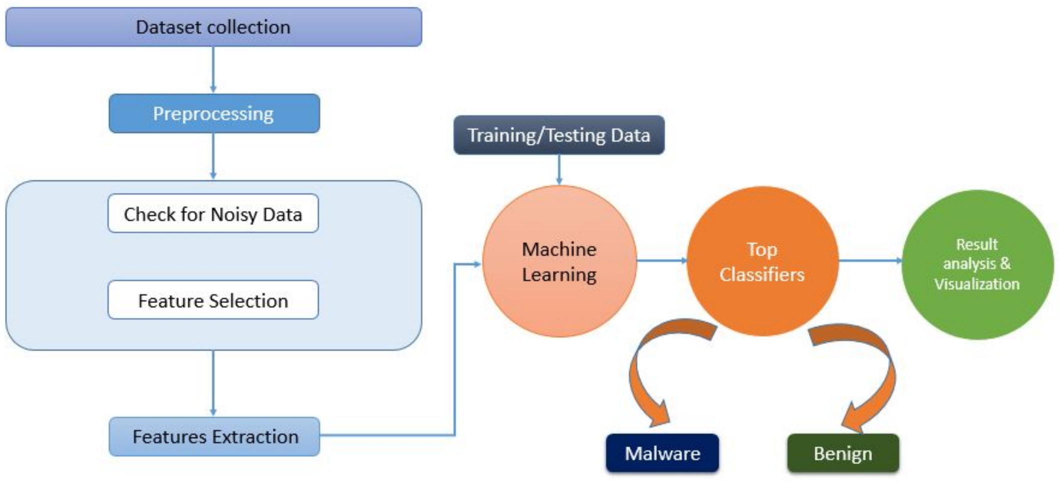


Figure 3.1: Workflow of the System

The workflow of our approach is given above to distinguish obscure malware by using machine learning. Our approach includes several steps to make it more accurate, efficient and effective. Our approach includes preprocessing of the dataset, feature selection, Training/Testing data in machine learning classification algorithm to detect malware or benign. After applying the classification algorithm we analyzed and visualized the result.

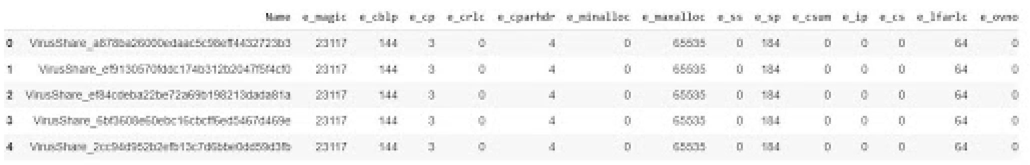
3.1.1 Dataset

Machine learning heavily depends on data. It is the most crucial aspect and the dataset makes the machine learning training feasible. The dataset is used to train the model for performing various actions, to work automatically. The training dataset is a dataset in which machine learning algorithm has trained and the dataset we use to validate the accuracy of our model is called testing dataset. Dataset can be built, downloaded from the website. Our dataset was collected from Microsoft Kaggle which contains nineteen thousand six hundred eleven samples. The specialty of our dataset is too many features. Our dataset contains seventy-nine features.

3.1.2 Data preprocessing

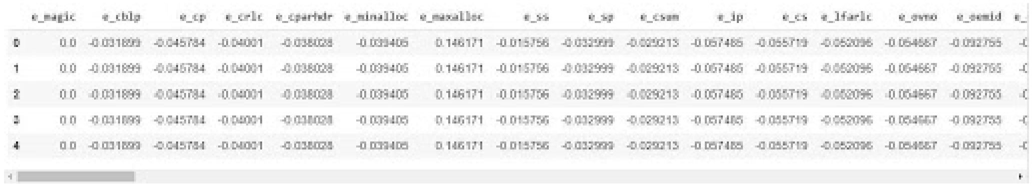
Data preprocessing is an information mining strategy that is used to change the unrefined information in an accommodating and appreciable format. One of the data preprocessing step is PCA (Principal Component Analysis) which we used in our approach. PCA is a dimension reduction technique to avoid over-fitting. It takes a dataset and "rotates" it, taking the original axes characterized by the original factors, and making new axes that are linear combinations of the old data. The linear combination exact is picked with the end goal that each progressive segment amplifies fluctuation along that new dimensions. It is a method of summarizing the data.

Another preprocessing step we used in the dataset is that we used standard scalar. Standardization of a dataset is a typical necessity for some, machine learning estimators. They may carry on severely if the individual features do not basically look like standard usually scattered information, For example, numerous elements utilized in the target capacity of a learning algorithm accept that all features are based on zero and fluctuation in a similar request. If a component has a change that is requests of extent bigger than others, it may overwhelm the target capacity and make the estimator unfit to gain from different features accurately true to form. It is useful for which has negative data. It will transfer data that its distribution will have standard deviation of one and mean value zero.



	Name	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csum	e_ip	e_cs	e_lfarlc	e_ovno
0	VirusShare_a878ba26000edaac5c98ef14432723b3	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
1	VirusShare_ef9130570fddc174b312b2047f5f4cf0	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
2	VirusShare_ef84cdeba22be72a69b198213dada81a	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
3	VirusShare_6bf3608e60ebc16cbcff6ed5467d469e	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0
4	VirusShare_2cc94d952b2efb13c7d6bbe0dd59d3fb	23117	144	3	0	4	0	65535	0	184	0	0	0	64	0

Figure 3.2: Before Data Preprocessing

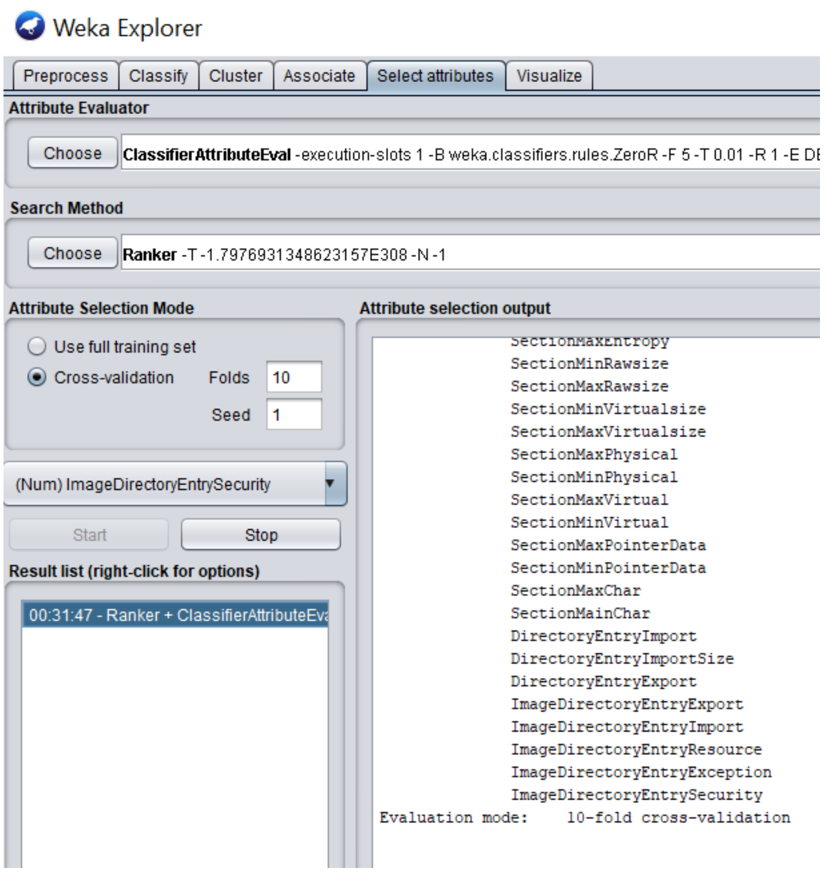


	e_magic	e_cblp	e_cp	e_crlc	e_cparhdr	e_minalloc	e_maxalloc	e_ss	e_sp	e_csum	e_ip	e_cs	e_lfarlc	e_ovno	e_oemid	e_
0	0.0	-0.031899	-0.045784	-0.04001	-0.038028	-0.039405	0.146171	-0.015756	-0.032999	-0.029213	-0.057485	-0.055719	-0.052096	-0.054667	-0.092755	-
1	0.0	-0.031899	-0.045784	-0.04001	-0.038028	-0.039405	0.146171	-0.015756	-0.032999	-0.029213	-0.057485	-0.055719	-0.052096	-0.054667	-0.092755	-
2	0.0	-0.031899	-0.045784	-0.04001	-0.038028	-0.039405	0.146171	-0.015756	-0.032999	-0.029213	-0.057485	-0.055719	-0.052096	-0.054667	-0.092755	-
3	0.0	-0.031899	-0.045784	-0.04001	-0.038028	-0.039405	0.146171	-0.015756	-0.032999	-0.029213	-0.057485	-0.055719	-0.052096	-0.054667	-0.092755	-
4	0.0	-0.031899	-0.045784	-0.04001	-0.038028	-0.039405	0.146171	-0.015756	-0.032999	-0.029213	-0.057485	-0.055719	-0.052096	-0.054667	-0.092755	-

Figure 3.3: After Data Preprocessing

Noisy Data: Noisy data is pointless data. The term has often been used as a comparable word for degenerate data information. Be that as it may, its significance has reached out to incorporate any data that can not be understood and interpreted effectively by machines, for example, unstructured content[26]. Any information that has been received, stored, or changed in such a way, that it can't be perused or utilized by the program that initially made it tends to be depicted as noisy. Noisy data pointlessly build the measure of extra space required and can likewise unfavorably influence the aftereffects of any data mining investigation. We checked for any noisy data in our dataset and we found nothing.

3.1.3 Feature selection



Attribute Evaluator

ClassifierAttributeEval -execution-slots 1 -B weka.classifiers.rules.ZeroR -F 5 -T 0.01 -R 1 -E DE

Search Method

Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode

Use full training set [✓ Cross-validation] Folds 10 Seed 1

(Num) ImageDirectoryEntrySecurity

Result list (right-click for options)

00:31:47 - Ranker + ClassifierAttributeEva

Attribute selection output

```
                SectionMaxEntropy
                SectionMinRawsize
                SectionMaxRawsize
                SectionMinVirtualsize
                SectionMaxVirtualsize
                SectionMaxPhysical
                SectionMinPhysical
                SectionMaxVirtual
                SectionMinVirtual
                SectionMaxPointerData
                SectionMinPointerData
                SectionMaxChar
                SectionMainChar
                DirectoryEntryImport
                DirectoryEntryImportSize
                DirectoryEntryExport
                ImageDirectoryEntryExport
                ImageDirectoryEntryImport
                ImageDirectoryEntryResource
                ImageDirectoryEntryException
                ImageDirectoryEntrySecurity
Evaluation mode:    10-fold cross-validation
```

Figure 3.4: Top Feature Selection in Weka

Feature Selection is where you consequently or physically select those Features which contribute most to your prediction variable or yield in which you are keen on. Having unimportant Features in data can minimize the accuracy of the models and cause the model to learn reliant on immaterial Features. Feature selection can reduce over-fitting, training time[27]. It also improves the accuracy of the algorithm. However, more features mean better accuracy. In our data-set, we have seventy-nine features. Among them, we worked with seventy-seven features. We dropped the target column and another column "Name".

Weka which stands for Waikato Environment for Knowledge Analysis is a collection of information analysis and visualization tools, exhibited by the University of Wakito.In weka in the attribute evaluator, we used ClassifierAttributeEval and for the searching method, we used Ranker. Ranker ranks each attribute and returns a score. For attribute selection mode we used 10 fold cross-validation. However,we used all the features of our work.

3.1.4 Training/Testing of Machine learning Classifier

After the feature selection process, the next step is training and testing of machine learning classifiers. We split our dataset into training and testing sets. After that, we train and test the dataset in some classification algorithms. Training data are used to fit and tune the models. Test data are represented as unseen data to evaluate the models[28]. Test data is utilized to perceive how well the machine can foresee new answers dependent on training. In our dataset, we used eighty percent data as training data and twenty percent as testing data. After splitting the dataset we train our model. We have used nine classification algorithms for our approach. The nine classifiers are

- Logistic Regression
- K-Nearest Neighbor (KNN)
- Random Forest
- Adaboost
- Support Vector Machine (SVM)
- Decision Tree
- XGBoost
- Artificial Neural Network (ANN)
- Extra Tree Classifier.

These classification algorithms were used with particular laws to detect whether it is malware or benign.

3.1.5 Result Analysis and Visualization

After receiving the outcome they were analyzed and visualized. As we used nine classification algorithm we analyzed their accuracy, Precision, Recall, True Positive Rate (TPR), False Positive Rate (FPR), f1-Score, Support, Confusion Matrix. For the visualization part, we have used Heatmap, Distribution Graph, Correlation Matrix, Pie plot, Counterplot.

3.2 Real-Time Implementation

We tried to apply machine learning techniques in real life to find real-time execution. For malware detection using machine learning, it is important to detect any malware in real life. It is also important to find computation time as well. To apply the real-time implementation of our approach we tried to build a client-server model using Flask.

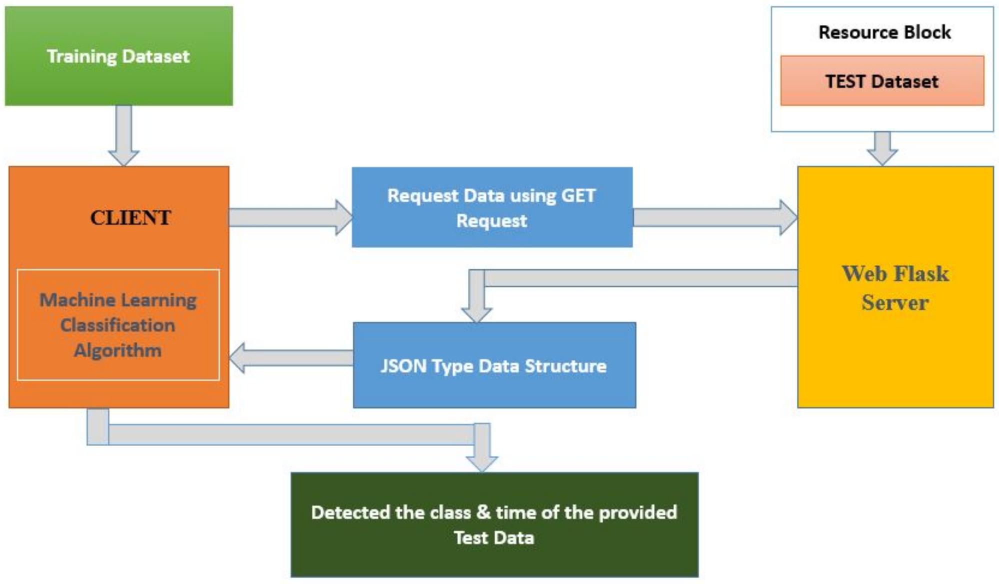

Training Dataset
Resource Block
TEST Dataset
CLIENT
Request Data using GET Request
Web Flask Server
Machine Learning Classification Algorithm
JSON Type Data Structure
Detected the class & time of the provided Test Data

Figure 3.5: Workflow of Real-time Implementation

Fig shows the workflow of the model to find real-time implementation to detect the malware. We built a client-server model for this approach. The process is given below:

3.2.1 Set up a flask local server

At the very first we set a local server for the model using Flask. A server means a medium or person that serves something based on requests. It provides data basically. So we built a server using the flask.

Flask: Basically, the flask is not a server. It is a web development framework. It is categorized as a micro-framework. It provides basic functionality to build a web application. However, it can be extended to more advanced operations. It is easy to learn and simple to use. In the flask, it is in the user's hand. One can do anything in it. There are no restrictions on the architecture or in the application layer. It is a developing network that is giving perfect answers for a few issues being looked at by engineers. To use this first we need to import flask. Every flask must need to create an application instance. After that, we used the flask-RESTful library here.

RESTful Library: In recent years REST has been recognized as the standard architectural design for web services. REST is designed by six design rules. These are:

- Client-server: There ought to be a detachment between the server that offers help and the client that devours it.
- Stateless: Each request from a client must contain all the data required by the server to do the request. At the end of the day, the server can't store data gave by the client in one request and use it in another request.
- Cacheable: The server must show to the client if requests can be reserved or not.
- Layered System: Communication between a client and a server ought to institutionalize so that enables delegates to react to demands rather than the end server, without the client doing anything extraordinary.
- Uniform interface: The strategy for correspondence between a client and a server must be uniform.
- Code on demand: Servers can give executable code or content to clients to execute in their unique circumstances. This imperative is the one in particular that is discretionary.

In the resource block, we uploaded our test dataset on the server. The test dataset is used basically to predict the unseen data. To measure the performance of the machine learning classification algorithm.

3.2.2 Client requests data from the server

The client means a person or organization using a specific service. The client-side requests data from the server by using a GET request. Here GET requests to include all required data. It can be bookmarked. GET requests can be re-executed. It is visible to everyone and it can be cached.

3.2.3 Response from server

After getting the requests from the client the flask server. After that, the server responds to the client. While responding the data is converted into JSON type data structure.

JSON Type Data: JSON (JavaScript Object Notation) is the most generally utilized data position for data exchange on the web. JSON is a lightweight text-based, data exchange organization and it totally language free. It depends on a subset of the JavaScript programming language and it is straightforward and productive. JSON supports mainly six types of data.

1. String
2. Number
3. Boolean
4. Object
5. Null
6. Array

3.2.4 Training machine learning classifier

On the client-side, we at first trained our machine learning classification algorithms with the training dataset. The algorithms were trained with the training dataset. After that when the clients received test data from the server which was converted into JSON data type the trained classifier in the client-side executed the test dataset.

3.2.5 Detect the malware and execution time

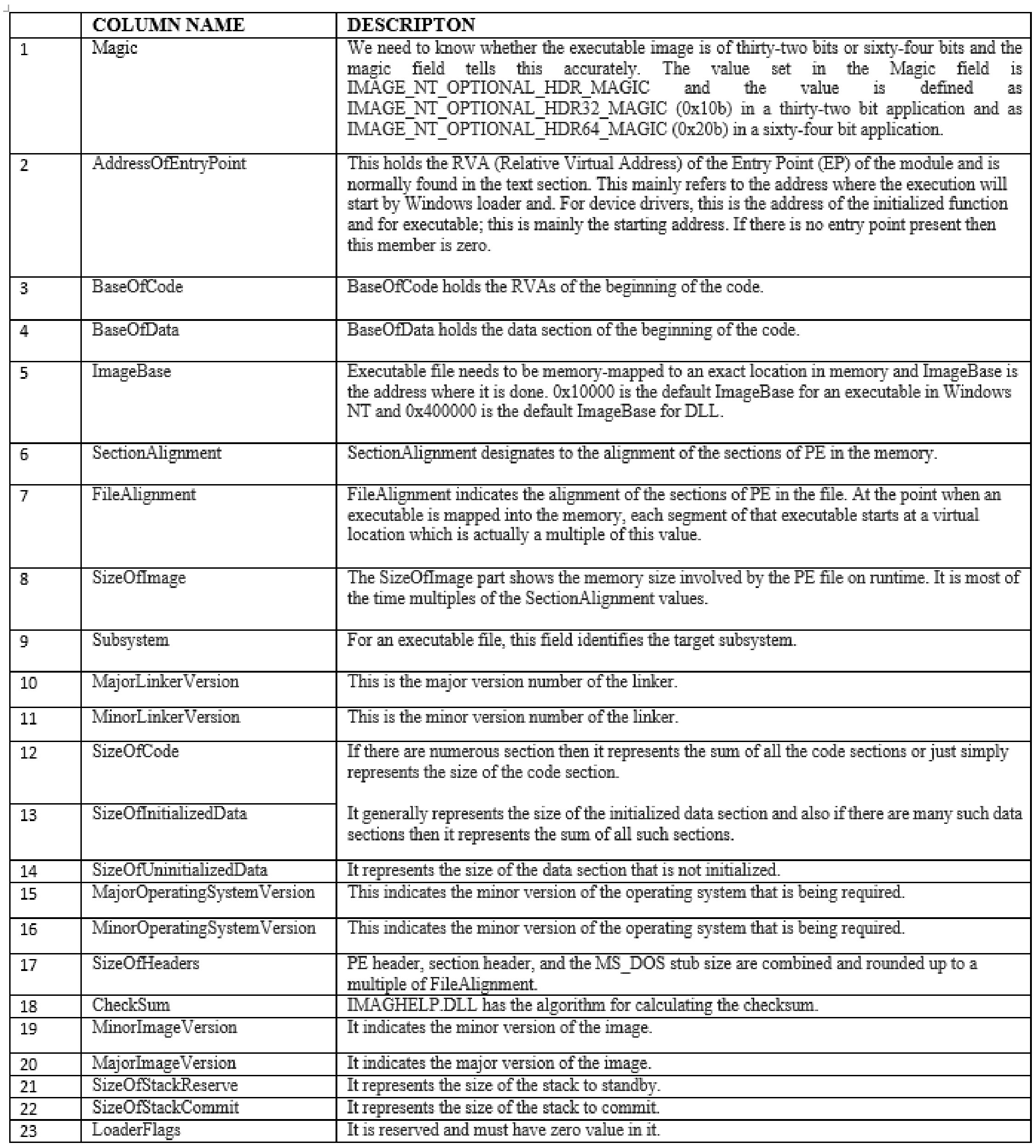
After getting the test dataset, the trained algorithm on the client-side executed it and detected the class of the test data. It also computed the execution time as well.

Chapter 4

Dataset description

Each executable document has a typical arrangement called Common Object File Format(COFF), an organization for executable, object code, shared library PC records utilized on Unix frameworks [29]. Also, PE(Portable Executable) design is one such COFF format accessible today for executable, object code, DLLs, FON font documents, and core dumps in thirty-two bit and sixty-four-bit versions of Windows operating systems. PE header actually holds the required data for the operating system to run the installation files like exe type data[30]. This comprises dynamic library references for linking, API export, and import tables, resource management data and TLS data. The Data Structures that are on the disk are the exact data structures that are being used in the memory. Instead of direct mapping of PE files into memory as a single memory-mapped file, the Win32 loader decides which portions of the PE files need to be mapped [29]. Our dataset is from kaggle and they used python pefile module to generate information of pe header and sections of pe file. We have used seventy-seven columns as input because the column “name” for input has string values and the column “malware” is our target or the output column. Out of all columns, forty-two columns have been described precisely below:

Table 4.1: Data set Description



	COLUMN NAME	DESCRIPTON
1	Magic	We need to know whether the executable image is of thirty-two bits or sixty-four bits and the magic field tells this accurately. The value set in the Magic field is IMAGE_NT_OPTIONAL_HDR_MAGIC and the value is defined as IMAGE_NT_OPTIONAL_HDR32_MAGIC (0x10b) in a thirty-two bit application and as IMAGE_NT_OPTIONAL_HDR64_MAGIC (0x20b) in a sixty-four bit application.
2	AddressOfEntryPoint	This holds the RVA (Relative Virtual Address) of the Entry Point (EP) of the module and is normally found in the text section. This mainly refers to the address where the execution will start by Windows loader and. For device drivers, this is the address of the initialized function and for executable; this is mainly the starting address. If there is no entry point present then this member is zero.
3	BaseOfCode	BaseOfCode holds the RVAs of the beginning of the code.
4	BaseOfData	BaseOfData holds the data section of the beginning of the code.
5	ImageBase	Executable file needs to be memory-mapped to an exact location in memory and ImageBase is the address where it is done. 0x10000 is the default ImageBase for an executable in Windows NT and 0x400000 is the default ImageBase for DLL.
6	SectionAlignment	SectionAlignment designates to the alignment of the sections of PE in the memory.
7	FileAlignment	FileAlignment indicates the alignment of the sections of PE in the file. At the point when an executable is mapped into the memory, each segment of that executable starts at a virtual location which is actually a multiple of this value.
8	SizeOfImage	The SizeOfImage part shows the memory size involved by the PE file on runtime. It is most of the time multiples of the SectionAlignment values.
9	Subsystem	For an executable file, this field identifies the target subsystem.
10	MajorLinkerVersion	This is the major version number of the linker.
11	MinorLinkerVersion	This is the minor version number of the linker.
12	SizeOfCode	If there are numerous section then it represents the sum of all the code sections or just simply represents the size of the code section.
13	SizeOfInitializedData	It generally represents the size of the initialized data section and also if there are many such data sections then it represents the sum of all such sections.
14	SizeOfUninitializedData	It represents the size of the data section that is not initialized.
15	MajorOperatingSystemVersion	This indicates the minor version of the operating system that is being required.
16	MinorOperatingSystemVersion	This indicates the minor version of the operating system that is being required.
17	SizeOfHeaders	PE header, section header, and the MS_DOS stub size are combined and rounded up to a multiple of FileAlignment.
18	CheckSum	IMAGHELP.DLL has the algorithm for calculating the checksum.
19	MinorImageVersion	It indicates the minor version of the image.
20	MajorImageVersion	It indicates the major version of the image.
21	SizeOfStackReserve	It represents the size of the stack to standby.
22	SizeOfStackCommit	It represents the size of the stack to commit.
23	LoaderFlags	It is reserved and must have zero value in it.

24	NumberOfRvaAndSizes	This portrays the size and location of the number of data- directory entries that are in the remainder of the optional header.
25	SizeOfHeapCommit	The space of the local heap space to commit.
26	TimeDateStamp	It shows the date and time when the debug data was created.
27	AddressOfRawData	Relative to the image base it shows the address of the debug data when loaded.
28	PointerToRawData	The file pointer of the debug data.
29	e_cblp	It shows how many bytes have been used in the last page and with a special case of a full-page being signified by a value of zero.
30	e_cp	It shows the number of pages required to hold the file.
31	e_crlc	It represents the number of entries that are present in the relocation pointer table. In short, it represents the number of relocation items.
32	e_cparhdr	In terms of paragraphs, it identifies the actual size of the executable header. Within the executable file, it specifies the offset of the program compiled and linked image.
33	e_minalloc	It clearly indicates the minimum number of extra paragraphs that might be required to be assigned to begin execution. This number indicates the total size of any uninitialized data or stack segments which are connected at the ending of a program.
34	e_maxalloc	It clearly indicates the maximum number of extra paragraphs that are required to be assigned by the program before the execution of the program starts. The program is assigned with as much memory as possible if the request that have been made cannot be satisfied.
35	e_ss	The SS value gets initialized by this which is the paragraph address of the stack segment.
36	e_sp	Before the program is credited all the control the SP value has to be located into the SP register and e_sp initializes this SP value.
37	e_scum	It integrates the data within the file. It also stipulates the checksum of the contents of the executable file.
38	e_ip	In order to transmission control to the program IP value has to be loaded into the IP register and e_ip specifies the initial IP value.
39	e_cs	CS value needs to be placed in the CS register for transmitting control to the program and e_cs states the pre-located initial CS value.
40	e_oemid	Indicates the identifier for the OEM for e_oeminfo.
41	e_oeminfo	For a precise value of e_oeminfo it postulates the OEM information.
42	e_lfanew	For the new exe header it states the file address. In precise, it could be a-byte offset into the file where the PE file header is found. It is fundamental to utilize this offset to find the PE header within the file.

Chapter 5

Algorithms

A regression model figures the class enrollment likelihood for one of the two classifications in the data set [26]. Machine learning is the latest buzzword in our world. We are using machine learning in many sectors like face recognition in our social media, applications like Siri, disease recognition in the healthcare industry, prediction on weather, analyzing customer behavior, etc. It actually refers to the feeding process to the machine to act intelligently without explicit programming, analysis the previous data to do the prediction of new data. We are also using machine learning to detect malware. Here, we are using supervised machine learning which refers to the events when we have all information are labeled or in another way, we can say we know how all information is classified. Classification and regression are types of supervised learning where we aware of the classes of classification and values in regression. However, in the case of unsupervised learning, we are not provided historical data so we have to do association or clustering to make classes. Now, the algorithms we used are described below:

5.1 Logistic Regression

A logistic regression algorithm is used to predict discrete or categorical values. It is mainly a classification algorithm that is used in cases like fraud detection, email spam detection, etc. where we have to make decisions between yes and no. It predicts the likelihood of event of an occasion by fitting information into the logit function. A regression model figures the class enrollment likelihood for one of the two classifications in the data set [31]. However, the Logistic curve is not a straight curve like linear regression. It is known as the sigmoid curve and here probability.

$$probability = 1/1 - e^{-straightline} \tag{5.1}$$

where straightline= mx+c. This equation of probability ensures that the predictor will be between 0 and 1. There are some steps to do the math behind the function. To start with we have to figure out the likelihood that a perception has a place with class utilizing the logistic response function. We characterize a limit to arrange each given information esteems into one of the classes. In this algorithm, a threshold value is required where bigger values than the threshold are treated as probability 1 and smaller values of the threshold are treated as probability 0. There are different types of logistic regression. Among them some are:

- Binary Logistic Regression: Yes or No, these two possible outcomes are possible from this type of logistic regression.
- Multi-nomial Logistic Regression: It has three nominal categories. For example-hound, elephant, crocodile.
- Ordinal Logistic Regression: It has at least three ordinal classification, ordinal implying that the classes will be in a request. For example - Model client ratings (1-5).

To build a model with logistic Regression we need to follow some steps which are given below:

1. Reading Data.
2. Data Analysis.
3. Split the data into training and test sets.
4. Classification report (performance, accuracy, etc.).

In our work after preprocessing by standard scaler, we used the logistic regression classifier.

5.2 K-Nearest Neighbor (KNN)

One of the simplest machine learning algorithms is KNN that is used for both classification and regression models. To assess the strategy we take a gander at three significant angles:

1. Straightforwardness to translate yield
2. Count time
3. Predict power

One of the simplest machine learning algorithms is KNN that is used for both classification and regression models. At that point, it finds the closest k individuals for that point. How do we choose the factor k :

First, let us attempt to comprehend what precisely does K impact in the algorithm. Choosing the value of k is very important and the value for k will help in improving the accuracy of the model. If we keep the value of k higher than it will help to reduce the variance but it can lead to a situation where the mall patterns in data are missed out.

The implementation of KNN can be done by following some steps which are given below:

1. Load the information.
2. Initialize the estimation of k.
3. For getting the anticipated class, emphasize from 1 to add up to number of training information points
4. Calculate the difference between test information and each row of training information. Here we will utilize Euclidean separation as our separation metric since it's the most prominent technique. For example, if we have two points Point1(X1, Y1) and Point2(X2, Y2) then

$$di(Point1, Point2) = \sqrt{(Y2 - Y1)^2 + (X2 - X1)^2} \tag{5.2}$$

Different measurements that can be utilized are Chebyshev, cosine, and so on

1. Sort the decided separation in rising request reliant on separation esteems.
2. Get top k rows from the arranged array.
3. Get the most successive class of these rows.
4. Return the anticipated class.

Pros and Cons of KNN:

Pros:

1. Simple implementation.
2. Makes no prior assumption of the data.

Cons:

1. Predict time is very high as the model finds the distance with every data point.

5.3 Random Forest

We need to acknowledge what class or gathering a perception has a spot with which can be perceived utilizing characterization. Characterization is a noteworthy bit of machine learning. The capacity to adequately characterize observations is incredibly basic for imagining stuff like whether a page is guaranteed to inspect or not, offering little appreciation to whether a site is protected to click or not, paying little respect to whether a site is destructive or not, etc. Close to the most raised motivation behind the classifier, the chain of criticalness is the random forest classifier. Random Forest, like its name induces, is various decision trees joined into a single model. How Random forest capacities are by at first building trees and afterward accumulating them. The more data the better the outcome as we merge all the more learning. As all the decision trees are consolidated, the figures will be nearer to the imprint. When looking at is done with substitutions, it draws the training set for current trees. After it's fulfillment, around thirty-three percent of the cases are chosen not to take any other potentially detrimental action for the model. As trees are added to the forest, the running fair check of the classification can be dictated by using the out-of-sack data. Subtleties of variable hugeness can in like manner be interpreted from it. To feasibly make an estimate, a subset of features is considered by every choice in the timberland while restricting requests and approaches an arbitrary training set of information points. Not too bad assortment of the forest is extended by this and makes progressively incredible all-around figures. Central purposes of using random forest include dealing with both course of action and backslide issues, working with a categorical and fast and tenacious variable, normally handles missing regards, not requiring scaling and is less influenced by noise. Both classification and regression tasks can be settled using Random Forest figuring. It won't overfit the model if we use more trees. The pseudocode of how random forest works is given below [32]:

1. Randomly select "K" features from complete "m" features where k is way less than m.
2. Among the "K" features, compute the node "d" utilizing the best split point.
3. Split the hub into daughter hubs utilizing the best split.
4. Repeat the 1 to 3 steps until "1" number of hubs has been come to Construct forest by repeating stages 1 to 4 for "n" number of times to make "n" number of trees.

The random forest can be used for both classification and regression. Probably the most serious issue in machine learning is overfitting, anyway, as a general rule this won't happen on account of the random forest classifier. If there are enough trees in the forest, the classifier won't overfit the model [32]. The fundamental confinement of random forest is that countless trees can make the algorithm excessively moderate and insufficient for continuous predictions.At the point when all is said in done, these algorithms rush to get ready, anyway exceptionally deferred to make expectations once they are prepared. A dynamically precise forecast requires more trees, which brings about an all the more moderate model. In most authentic applications, the random forest algorithm is sufficiently speedy yet there can without a doubt be conditions where run-time execution is noteworthy and various systems could be loved [32].

5.4 AdaBoost

Boosting, in general, is a sequential process where we create multiple weak classifiers into a big ensemble strong classifier. It helps to improve the learners by focusing on areas where the system is not performing well. One of the best algorithms in this sector is AdaBoost, also known as Adaptive Boosting. The algorithm mainly increases the weight of certain training data inputs by using ensemble learning. There are some steps on how AdaBoost performs.

1. First of all, it initializes weights to all training points. At initialization, every point that is used to train the classifier will be assigned a weight which is equal to the reciprocal of the total number of points that are present in my training data set.

2. Then it calculates the error rate for each weak classifier that is present by growing multiple decision stumps and pick the decision stumps with the lowest error rate.

3. The voting power of the weak classifier is computed and then appended the classifier into the ensemble classifier.

4. The next step is to update the weight of every training point based on whether it was classified correctly or incorrectly by the previous classifier. The weight goes up for a point if it was classified incorrectly and goes down if it was classified correctly. This process is repeated multiple times to create the best possible AdaBoost classifier.

AdaBoost algorithms can be utilized to take care of both classification and regression issues[33]. Ensembles of decision stumps can be useful to generalized linear models and also adds the ability to see even non-linear relationships between individual features and labels. Real-world data do not always show linear relationship but ensembles of decision stumps help us to catch most of the non-linear relationship which helps to predict more accurately.

5.5 Support Vector Machines

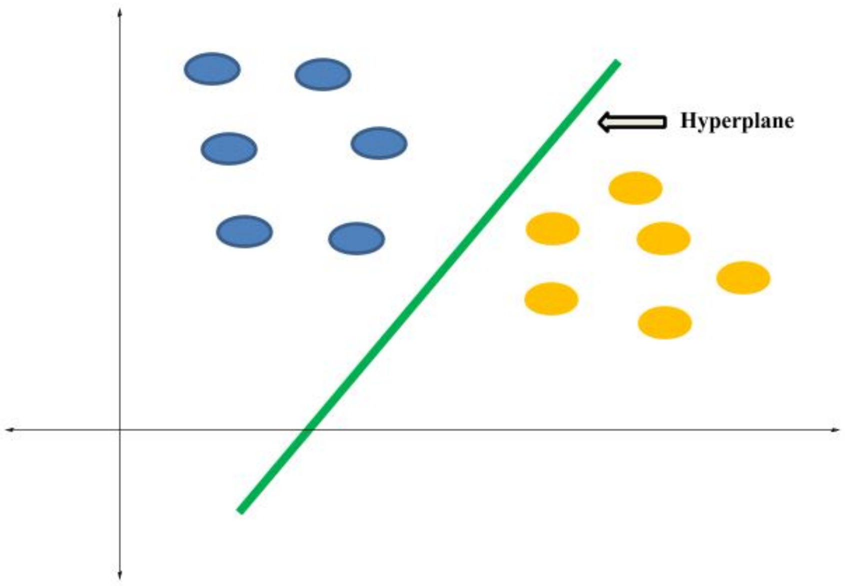


Figure 5.1: Support Vector Machine

Support Vector Machine is a supervised machine learning algorithm that can be used for both classification and regression. The goal of SVM is to find the hyperplane which divides the two classes of the data. Suppose, two classes are considered the training set inputspace of N pairs (xa, ya), a = 1,..., N, which is used during the training process can be defined as follows[34]:

$$inputspace = \{(x_a, y_a) | x_a \varepsilon R^n, y_a \varepsilon \{-1, 1\}\} \tag{5.3}$$

Where xa represents a training sample consisting of n values and yin is the binary level. Given a training dataset the SVM calculation scans for a plane (a hyperplane for n greater than 3) in the info space that isolates the positive examples from the negative ones[35]. In the original SVM model, all hyperplane is parameterized by a vector and constant

$$vector^T x + constant = zero \tag{5.4}$$

For hyperplane (vector, constant) we can formulate the classification rule[35]

$$h(x) = sign(vector^T x + constant) \tag{5.5}$$

Here h(x) will correctly classify the sample data.

Types of the kernel :

Polynomial:

$$k(x_a, x_b) = (\gamma x_a^T x_b + r)^d, \gamma > 0 \tag{5.6}$$

Radial basis function :

$$\exp(\gamma \left\| x_a x_b \right\|^2, \gamma > 0 \tag{5.7}$$

Sigmoid :

$$k(x_a, x_b) = \tan h \tan h(\gamma x_a^T x_b + r), \gamma > 0 \tag{5.8}$$

Pros of SVM:

1. It is compelling in high dimensional spots.
2. It works truly well with a reasonable edge of order.It is successful in situations where the quantity of measurements is more prominent than the quantity of tests.
3. It is successful in situations where the quantity of measurements is more prominent than the quantity of tests.

Cons of SVM:

1. It does not perform well when we have a very large data set.
2. Low performance if the data set is noisy.

5.6 Decision Trees

The decision tree algorithm can be utilized to take care of both classification and regression issues. We can speak to any boolean capacity on discrete characteristics utilizing the Decision tree. There are a few presumptions we have to settle on while utilizing the decision trees. At first, we need to consider the root hub as the training set. Feature values are liked to be straight out. On the off chance that the qualities are continuous, at that point, they are discretized preceding structure the model. Based on quality qualities records are conveyed recursively. We utilize factual strategies for requesting qualities as root or the inside hub. In Decision Tree, the significant test is to recognizable proof of the characteristic for the root hub in each level. This procedure is known as attribute selection. We have two famous property choice measures:

1. Information Gain.
2. Gini Index.

Information Gain: From[36], assume S is a set of examples, A is a property.Split the training set into subsets. Ensure every subset contains information with a similar incentive for a characteristic. Sx is the subset of S with A = x, and Value (A) is the arrangement of every estimation of A, at that point.

$$Gain(S, A) = Entropy(S) - \sum ValueA.\frac{Sx}{s}.Entropy(Sx) \tag{5.9}$$

Here Entropy is the measure of the vulnerability of an irregular variable, it describes the polluting influence of a self-emphatic variety of examples. The higher the entropy more the information content. So to decide the attribute we use info gain to choose which attribute to level with each hub.

Gini Index: Gini Index is an estimation to measure how routinely a discretionarily picked segment would be incorrectly perceived. It infers a characteristic with a lower Gini record should be liked. Sklearn underpins the "Gini" criteria for Gini Index and as per normal procedure, it takes the "Gini" esteem. The recipe to compute the Gini Index is given underneath:

$$G.I = 1 - \sum_{a}^{\infty} p_a \tag{5.10}$$

Decision Tree Algorithm Pseudocode :

1. Spot the best quality of the dataset at the root node.Split the training set into subsets. Ensure every subset contains information with a similar incentive for a characteristic.
2. Split the training set into subsets. Ensure every subset contains information with a similar incentive for a characteristic.
3. Repeat step 1 and 2 until we find the final leaf node in all the branches of the tree.

Pros of Decision Tree :

1. Straightforward
2. Valuable in information investigation. .
3. Less information cleaning required.
4. Little effort from users for data preparation.

Cons of Decision Tree:

1. Over-fitting problems may occur.
2. Does not easily handle non-numeric data.
3. It can be quite large.
4. Pruning is necessary.

5.7 XGBoost

XGBoost is an algorithm that has as of late been commanding applied machine learning and Kaggle competitions for organized or unthinkable information. XGBoost is a utilization of gradient boosting decision trees planned for speed and execution. Three primary types of gradient boosting are supported:

1. Gradient Boosting algorithm likewise called gradient boosting machine which includes the learning rate.
2. Stochastic Gradient Boosting with sub-testing at the line, section,and frag- Inadequate Aware utilization with the modified treatment of missing data esteem.fragment per split levels.
3. L1 and L2 regularization can be used to regularize gradient boosting.

The execution of the algorithm was built for the effectiveness of process time and memory assets. A plan objective was to utilize accessible assets to train the model. Some key algorithm execution highlights include:

1. Inadequate Aware utilization with the modified treatment of missing data esteem.
2. Block Structure to help the parallelization of tree development.
3. L1 and L2 regularization can be used to regularize gradient boostingL1 and L2 regularization can be used to regularize gradient boosting

XGBoost modifies the boosting algorithm based on GBDT(Gradient Boosting Decision Trees). This algorithm is made out of numerous regression trees, and the last outcome is an added substance mix of the decision results of all subtrees.Regardless, the glaring prohibition of GBDT is the need to utilize the extra misstep of the n-1th tree when training the nth tree, which makes GBDT difficult to be passed on the spread structure. So as to address this issue, XGBoost utilizes Taylor expansion on the misfortune work[37]

$$\begin{aligned} lk(y_i, \overrightarrow{x_i}) &= lk(y_i, F_{k-1}\overrightarrow{x_i} + f_{k-1}\overrightarrow{x_i} \approx l(F_{k-1}\overrightarrow{x_i}) \\ &+ \frac{\partial l(y_i, F_{k-1}\overrightarrow{x_i})}{\partial(F_{k-1}\overrightarrow{x_i})} F_k\overrightarrow{x_i} + \frac{1}{2}\frac{\partial^2 l(y_i, F_{k-1}\overrightarrow{x_i})}{\partial(F_{k-1}\overrightarrow{x_i}^2)} F_k^2\overrightarrow{x_i}) \end{aligned} \tag{5.11}$$

Moreover, regular item is added to the loss function to avoid over-fitting which is given below

$$\Omega f_t = \gamma T + \frac{1}{2}\lambda \sum_{a=1}^{T} w_a^2 \tag{5.12}$$

For ample scanty discrete features, the XGBoost will possibly count the non-zero examples when searching for rending point[37]

5.8 Artificial Neural Network (ANN)

Artificial Neural Network Algorithm (ANN) is developed from the idea of how the neurons of human brains are connected, and how the nervous system performs its work. Artificial Neural Network has many uses and classification is one of them. It can perform well for large data-sets; instead of taking the entire data-set it takes data samples. Basically, the artificial neural network has three different layers, the first one is the input layer, where data are given as input, the second one is the hidden layer and the final one is the output layer. Depending on the number of hidden layers varies. ANN displaying begins with randomly appointed weight coefficients. At that point, a lot of information designs are nourished forward over and again, and the loads of the neurons are changed until the yield coordinates intimately with the real values. In our model, we used two hidden layers. We used Rectified Linear Unit (ReLU) and Sigmoid function as activation function in our model. After the model is defined we compiled it, we used cross-entropy as the loss function. For optimization, we used Adam optimization algorithm that is the extension of a greatly used optimization algorithm stochastic gradient descent.

5.9 Extra Tree Classifier

The Extra-Tree technique (representing extremely randomized trees) was proposed with the primary goal of further randomizing tree working with regards to numerical information features, where the decision of the ideal cut-point is answerable for an enormous extent of the fluctuation of the incited tree. The Extra-Trees algorithm constructs a gathering of unpruned choice or relapse trees as per the traditional top-down strategy. Its two principal contrasts with other tree-based outfit techniques are that it parts nodes by picking cut-focuses completely indiscriminately and that it utilizes the entire learning test (as opposed to a bootstrap copy) to develop the trees. The Extra-Tree has two parameters: K, the number of properties arbitrarily chose at every node and nmin, the least example size for splitting a node. It is utilized a few times with the (full) unique learning test to create a troupe model (we mean by M the quantity of trees of this group). The forecasts of the trees are collected to yield the last expectation, by the greater part vote in characterization issues and number juggling normal in relapse issues. From the predisposition change perspective, the justification behind the Extra-Trees strategy is that the unequivocal randomization of the cut-point and property joined with gathering averaging ought to have the option to diminish change more emphatically than the more fragile randomization plans utilized by different strategies. The use of the full unique learning test as opposed to bootstrap copies is persuaded to limit predisposition. From a computational perspective, the unpredictability of the tree developing technique is, expecting adjusted trees, on the request for N log N as for learning test size, like most other tree developing systems. Be that as it may, given the effortlessness of the hub parting system we anticipate that the consistent factor should be a lot littler than in other troupe based techniques which locally advance cut-focuses. The parameters K, nmin and M have various impacts: K decides the quality of the quality determination process, nmin the quality of averaging yield clamor, and M the quality of the change decrease of the gathering model total. These parameters could be adjusted to the issue points of interest in a manual or a programmed way (for example by cross-approval). In any case, we want to utilize default settings for them to boost the computational points of interest and self-governance of the strategy. Segment 3 examinations these default settings as far as power and suboptimality in different settings. To indicate the estimation of the principle parameter K, we will utilize the documentation ETK, where K is supplanted by 'd' to state that default settings are utilized, by star to signify the best outcomes acquired over the scope of potential estimations of K, and by 'cv' if K is balanced by cross-approval.

Chapter 6

Result analysis and Data Visualization

We set up a server and a client with the help of the Python Flask. On the client-side, we implemented our machine learning classifier. The data got by the client is passed to the trained classifier, the trained classifier then predicts the label of the data. In our case, it took around 8 seconds for the classifier to classify the data in real-time. The following figure shows the output of the classifier on the client-side.

6.1 Data visualization

In order to work with any dataset, it is very important to visualize it, python has a bunch of libraries with the help of which we can easily visualize the data. We used python libraries like seaborn, mathplotlib, scikitplot, etc to plot heatmap, bar graph, correlation matrix, scatter and density, count plot, pieplot, etc.

6.1.1 Heat map

We used the python sea-born library to plot the heatmap. Heatmap is actually a colored tabular matrix where the color of each cell depends on the data contained in it. It shows the correlation.

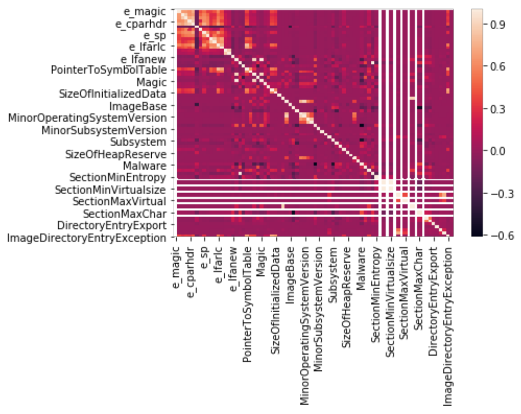


Figure 6.1: Heatmap

6.1.2 Bar graph

The bar graph is used to show the distribution of numerical data. It is one of the most used methods for showing statistical data. In the bar graph, rectangular bars are used to represent data. The bar of different heights shows the frequency of different items. We plotted the distribution of the first twenty columns.

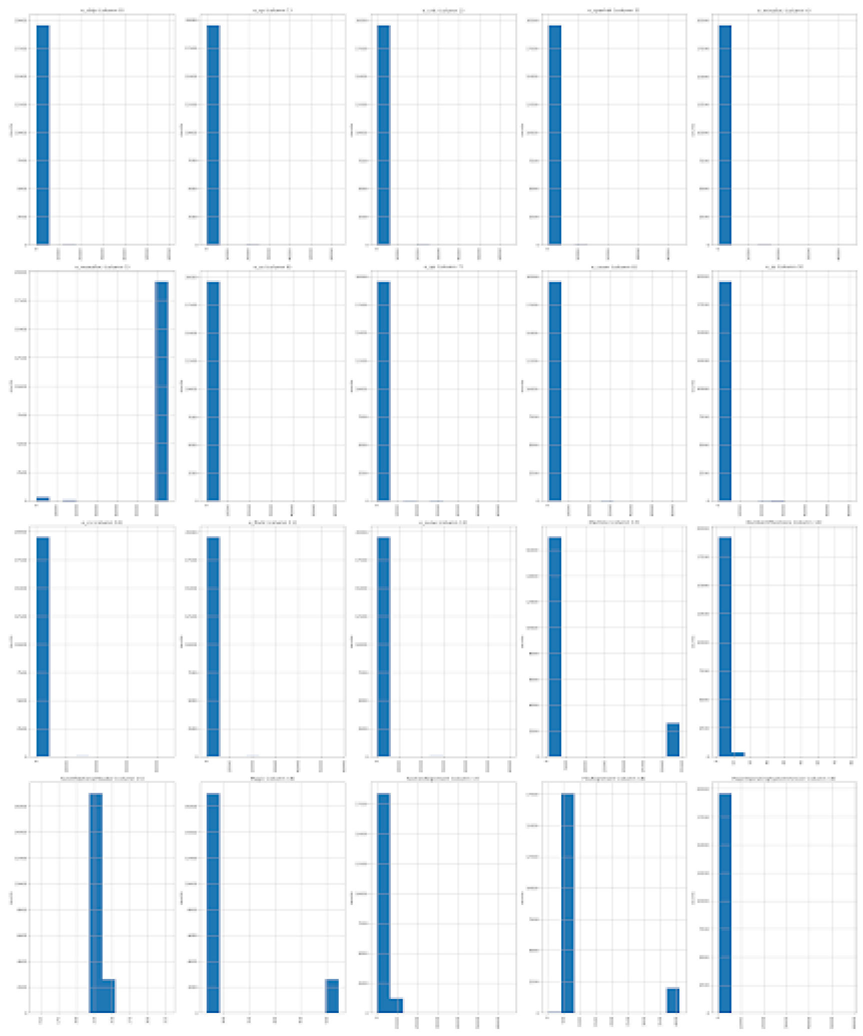


Figure 6.2: Bar Graph

6.1.3 Correlation Matrix

A correlation matrix is a matrix that shows the correlation coefficient among the set of variables. The values of correlation are standardized, that is the value can range from -1 to +1. We plotted the correlation matrix of our data, we used the matplotlib library of Python for plotting the matrix.

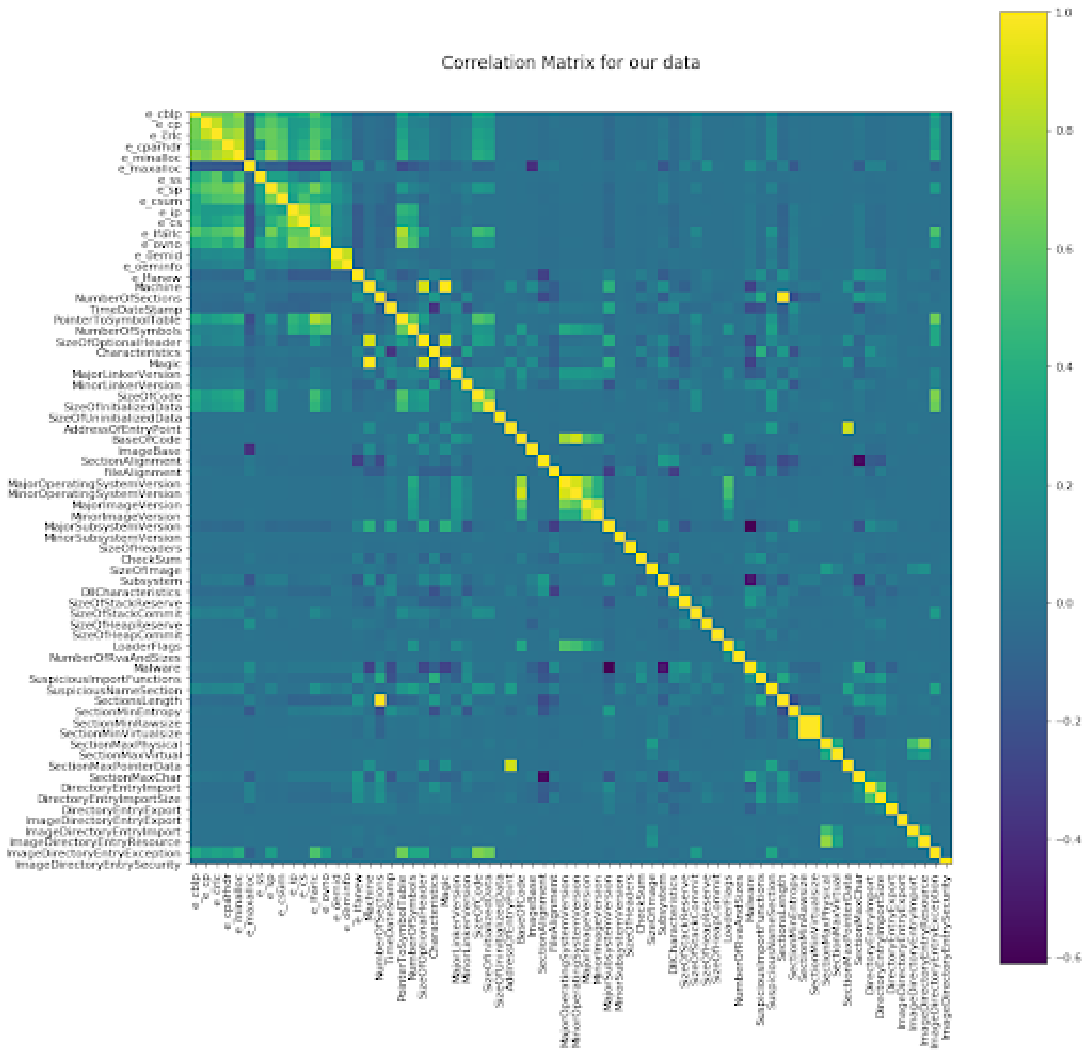


Figure 6.3: Correlation Matrix

6.1.4 Count plot

A count plot is used to visualize the number of items present in each category in the form of a rectangular bar. We used the seaborn visualization libraries count plot to visualize our data. In our data 0 stands for benign and 1 stand for malware.

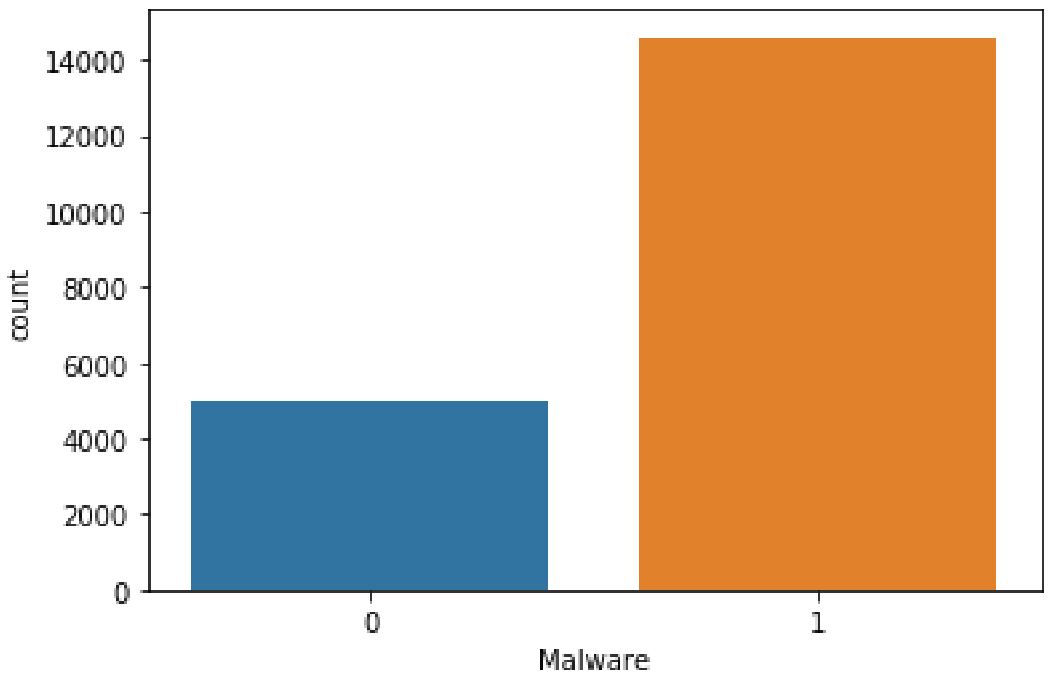


Figure 6.4: Count Plot

6.1.5 Pie plot

Pie plot is one of the most widely used forms of visualizing data, it is used to plot the pie chart. In the pie chart, the percentage of data in each category is represented as a slice of the circle. With the help of pie chart statistical data can be easily represented. Here in the pie plot, 0 represents benign and 1 represents malware data.

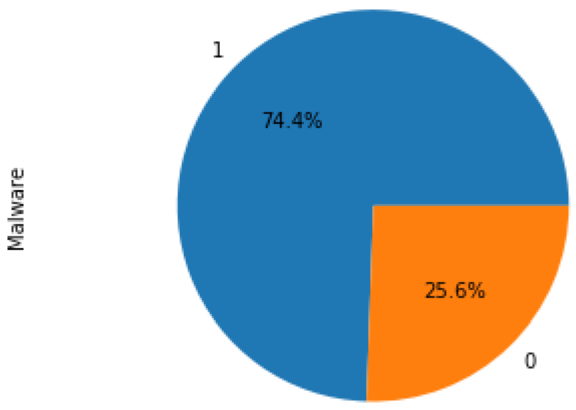


Figure 6.5: Pieplot

6.2 Result analysis

In this part, we are going to analyze the result with the help of different evaluation metrics such as classification report, confusion matrix, roc curve, accuracy score, etc.

We used nine classifiers to classify the malware and benign data, they are Logistic Regression, K-Nearest Neighbor (k-NN), Random Forest, AdaBoost, Support Vector Machines (SVM), Decision Trees, XGBoost, Artificial Neural Network (ANN) and Extra Tree Classifier. After preprocessing we split the data into training and test set. We trained the classifier with the training set and then we tested them using the test data. We evaluated the performance of the classifiers with the help of different metrics. Different metrics used by us for evaluation are discussed as follows :

Confusion Matrix:

One of the most widely used methods of evaluating the machine learning algorithm is the confusion matrix.

Table 6.1: Table Of Confusion Matrix

Confusion Matrix	Negative(Predicted)	Positive(Predicted)
Actually Negative	T.N	F.P
Actually Positive	F.N	T.P

Here, True Negative (T.N) means that the algorithm predicted negative and the result is actually negative, False Positive (FP) means that the algorithm predicted positive but the result is actually negative, False Negative (FN) means that the algorithm predicted negative but the result is actually positive and True Positive (TP) means that the algorithm predicted positive and the result is actually positive.

True Positive Rate (T.P.R): TPR is also known as recall. It shows the ability of our classifier to detect malware

$$T.P.R = TruePositive/(TruePositive + FalseNegative) \tag{6.1}$$

False Positive Rate (F.P.R): FPR shows that the possibility of benign files wrongly classified as malware

$$F.P.R = FalsePositive/(FalsePositive + FalseNegative) \tag{6.2}$$

Classification Report: The classification report is another evaluation metrics for the evaluation machine learning algorithm. We used the python sklearn libraries classification report to show the classification report of different algorithms. With the help of the classification report, we calculate the parameters like Precision, Recall, F1 score, Support, etc.

- Precision: Precision is the ratio of correctly predicted positive to the total predicted positive sample. The formula for prediction is:

$$Precision = TruePositive/TruePositive + FalsePositive \tag{6.3}$$

- Recall: Recall is the ability of an algorithm to find all positive samples. The formula for the recall is:

$$Recall = TruePositive/TruePositive + FalseNegative \tag{6.4}$$

- F1 score: F1 Score is the weighted average of Precision and Recall. The formula for F1 score is:

$$F1Score = 2 * (Recall * Precision)/(Recall + Precision) \tag{6.5}$$

- Support: The support is the number of samples of the true sample that lie in that class i.e. number of occurrences of each class.
- Micro average: We get micro average by averaging the total true positives, false negatives and false positives samples.
- Macro average: Macro average calculates the metrics for each label, and find their unweighted mean.
- Weighted average: Weighted average calculates the metrics for each label, and find their average weighted by support.

ROC Curve

Receiver Operating Characteristic curve (ROC Curve) is used to plot the true positive rate against the false-positive rate. With the help of ROC Curve, we can find the capability of a classifier to differentiate between different classes. The Area Under Curve (AUC) is the area under the ROC Curve, with the help of AUC score we get the idea of how well the model is performing. We used the python sklearn library to find the ROC Curve and score for different algorithms.

Accuracy Score

Accuracy score is the most common metric for evaluating a model. We used sklearn Accuracy score metrics to find the accuracy score for different algorithms. The formula for finding the accuracy score is as follow :

$$AccuracyScore = \frac{T.P + T.N}{T.P + T.N + F.P + F.N} \tag{6.6}$$

Here, T.P= True Positive, T.N = True Negative,F.P= False positive, F.N= False Negative

The results obtained from the different algorithm is described as follows :

6.2.1 Logistic Regression

For logistic regression, we get an accuracy score of 0.963.

The confusion matrix for logistic regression is as follows :

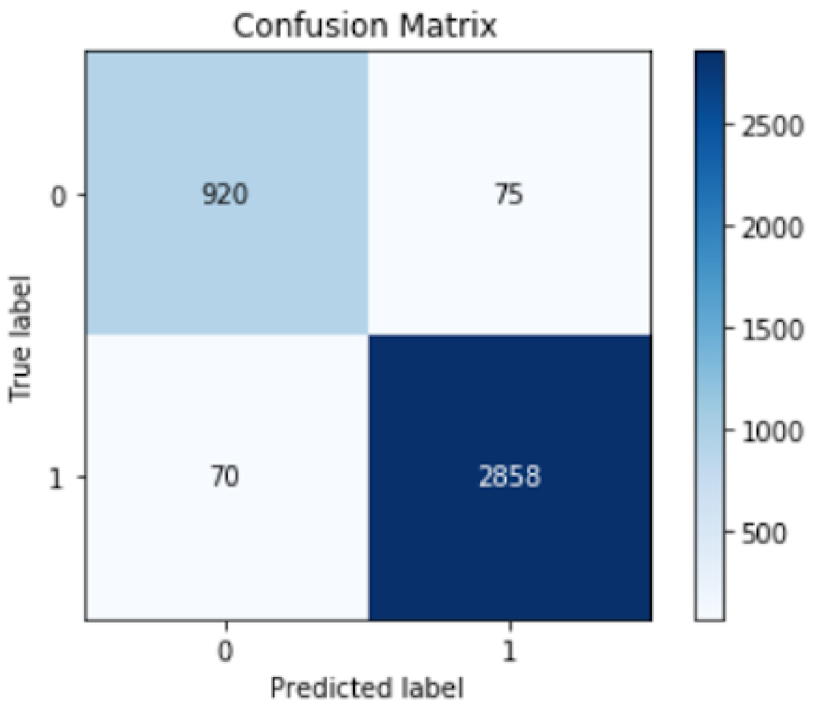
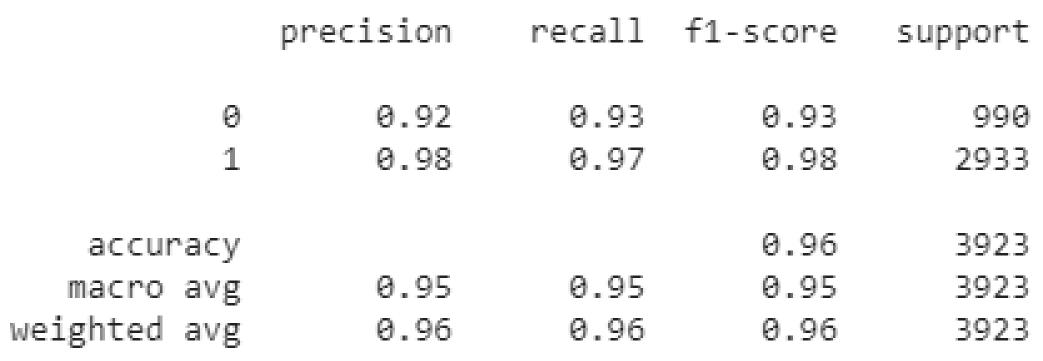


Figure 6.6: Classification Matrix of Logistic Regression

Here in the confusion matrix, the true negative value is nine hundred twenty, the false positive value is seventy-five, the false-negative value is seventy and the true positive value is two thousand fifty-eight.

The classification report for logistic regression is as follows :



```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93       990
           1       0.98      0.97      0.98      2933

    accuracy                           0.96      3923
   macro avg       0.95      0.95      0.95      3923
weighted avg       0.96      0.96      0.96      3923
```

Figure 6.7: Classification Report of Logistic Regresion

In the classification report of logistic regression, we can see that the precision, recall, and f1-score for benign is 0.92,0.93 and 0.93 respectively. For malware, the precision, recall, and f1-score are 0.98,0.97 and 0.98 respectively. Here the number of benign files detected is nine hundred ninety and the number of malignant files detected is two thousand nine hundred thirty-three.

The roc curve for logistic regression is as follows :

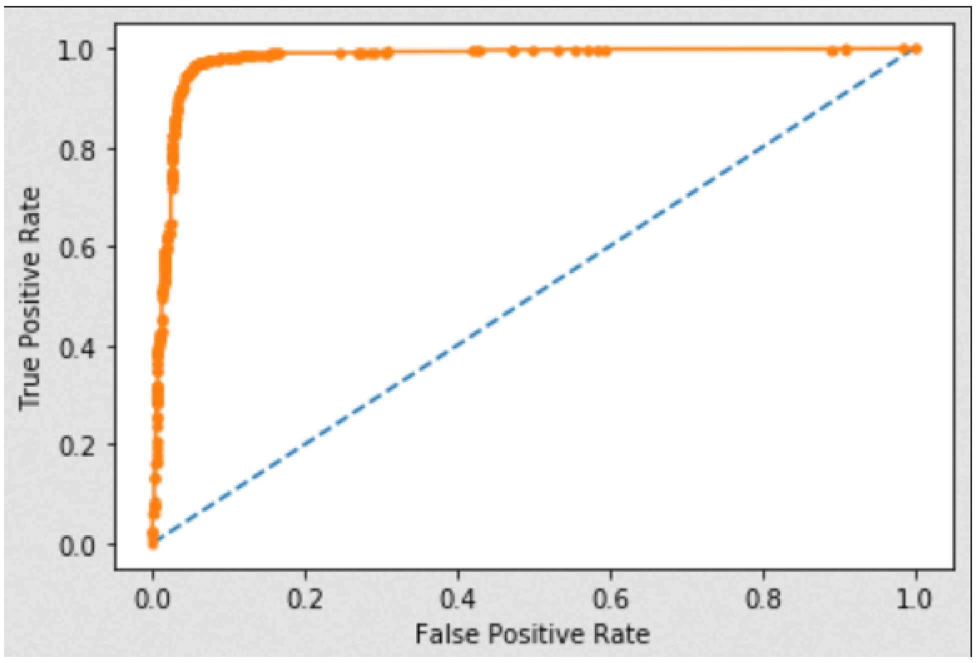


Figure 6.8: ROC of Logistic Regression

For logistic regression we get the AUC(Area Under Curve) score 0.978.

6.2.2 K-Nearest Neighbor (k-NN)

For K-Nearest Neighbor (k-NN) we get an accuracy score of 0.959.

The confusion matrix for K-Nearest Neighbor (k-NN) is as follows :

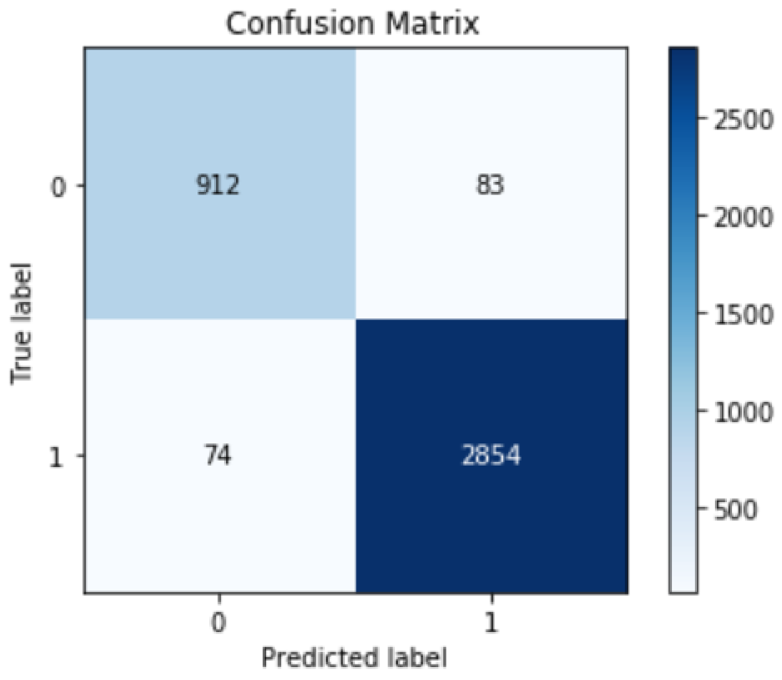
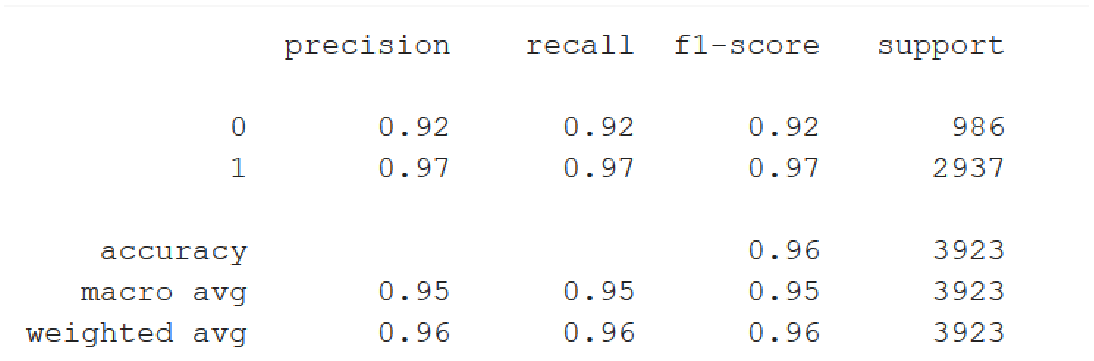


Figure 6.9: Classification Matrix of KNN

Here in the confusion matrix, the true negative value is nine hundred twelve, the false positive value is eighty-three, the false-negative value is seventy-four and the true positive value is two thousand fifty-four.

The classification report for K-Nearest Neighbor (k-NN) is as follows :



```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       986
           1       0.97      0.97      0.97      2937

    accuracy                           0.96      3923
   macro avg       0.95      0.95      0.95      3923
weighted avg       0.96      0.96      0.96      3923
```

Figure 6.10: Classification Report of KNN

In the classification report of K-Nearest Neighbor (k-NN), we can see that the precision, recall, and f1-score for benign is 0.92,0.92 and 0.92 respectively. For malware the precision, recall, and f1-score are 0.97,0.97 and 0.97 respectively. Here the number of benign files detected is nine hundred eighty-six and the number of malignant files detected is two thousand thirty-seven.

The roc curve for K-Nearest Neighbor (k-NN) is as follows :

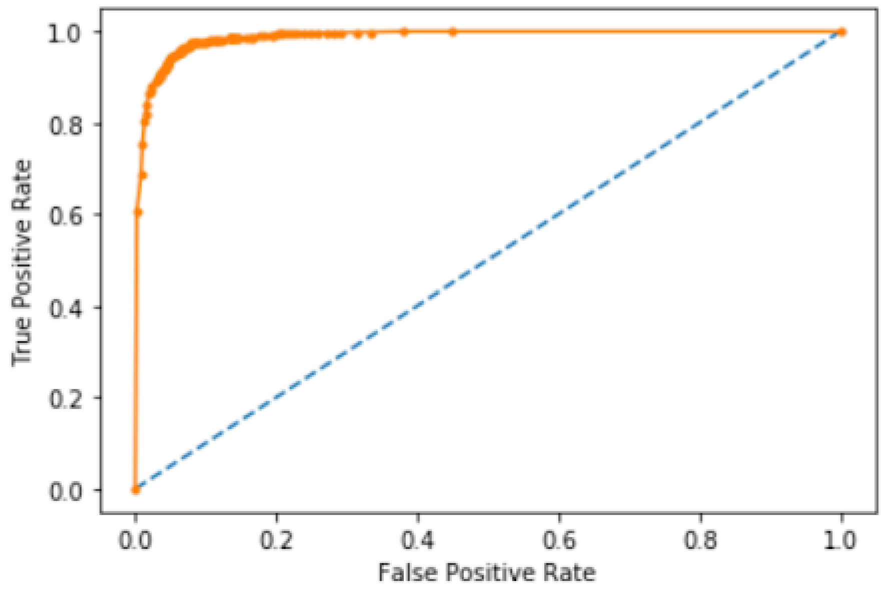


Figure 6.11: ROC of KNN

For K-Nearest Neighbor (k-NN) we get the AUC(Area Under Curve) score 0.987.

6.2.3 Random Forest

For Random Forest we get an accuracy score of 0.981.

The confusion matrix for Random Forest is as follows :

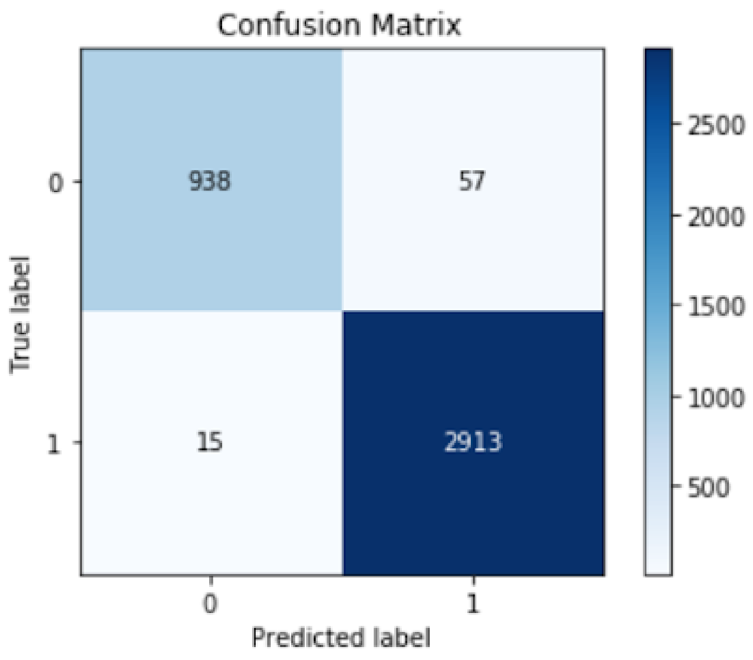
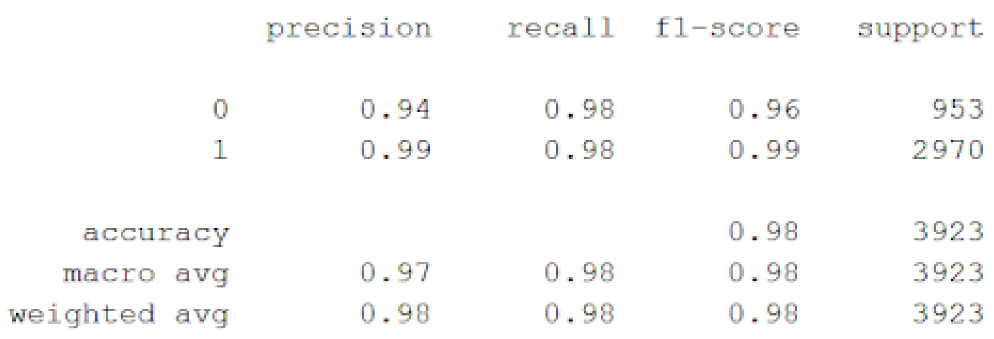


Figure 6.12: Classification Matrix of Random Forest

Here in the confusion matrix, the true negative value is nine hundred thirty-eight, the false positive value is fifty-seven, the false-negative value is fifteen and the true positive value is two thousand nine hundred thirteen.

The classification report for Random Forest is as follows :



```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       953
           1       0.99      0.98      0.99      2970

    accuracy                           0.98      3923
   macro avg       0.97      0.98      0.98      3923
weighted avg       0.98      0.98      0.98      3923
```

Figure 6.13: Classification Report of random forest

In the classification report of Random Forest, we can see that the precision, recall, and f1-score for benign is 0.94,0.98 and 0.96 respectively. For malware the precision, recall, and f1-score are 0.99,0.98 and 0.99 respectively. Here the number of benign files detected is nine-hundred fifty-three and the number of malignant files detected is two thousand seventy.

The roc curve for random forest is as follows :

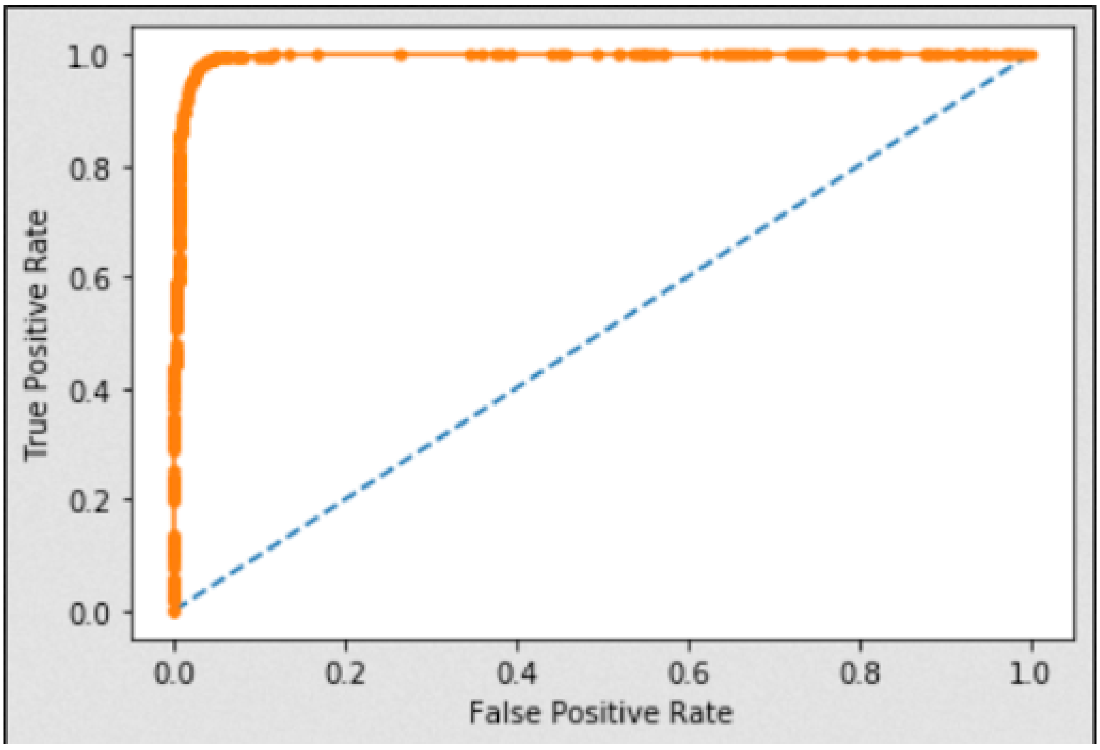


Figure 6.14: ROC of Random Forest

For Random Forest we get the AUC(Area Under Curve) score 0.995.

6.2.4 AdaBoost

For Adaboost, we get an accuracy score of 0.972.

The confusion matrix for AdaBoost is as follows :

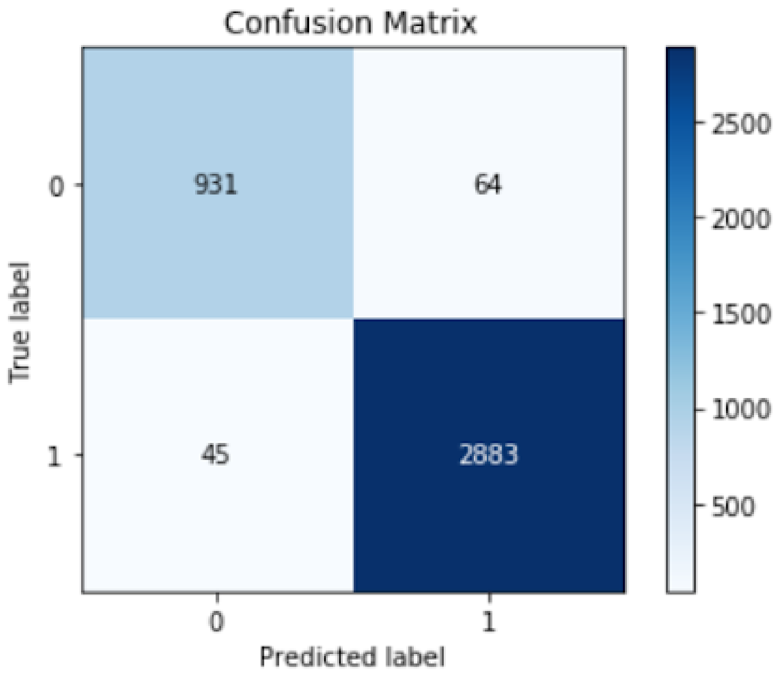
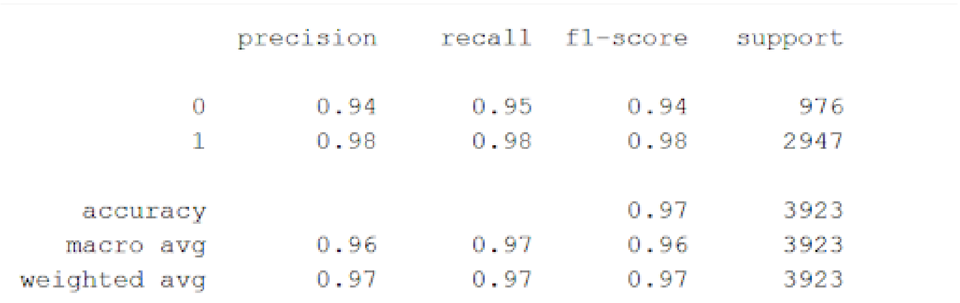


Figure 6.15: Classification Matrix of Adaboost

Here in the confusion matrix, the true negative value is nine hundred thirty-one, the false positive value is sixty-four, the false-negative value is forty-five and the true positive value is two thousand eight hundred eighty-three.

The classification report for AdaBoost is as follows :



```
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       976
           1       0.98      0.98      0.98      2947

    accuracy                           0.97      3923
   macro avg       0.96      0.97      0.96      3923
weighted avg       0.97      0.97      0.97      3923
```

Figure 6.16: Classification Report of AdaBoost

In the classification report of AdaBoost, we can see that the precision, recall, and f1-score for benign is 0.94,0.95 and 0.94 respectively. For malware, the precision, recall, and f1-score are 0.98,0.98 and 0.98 respectively. Here the number of benign files detected is nine hundred seventy-six and the number of malignant files detected is two thousand forty-seven.

The roc curve for AdaBoost is as follows :

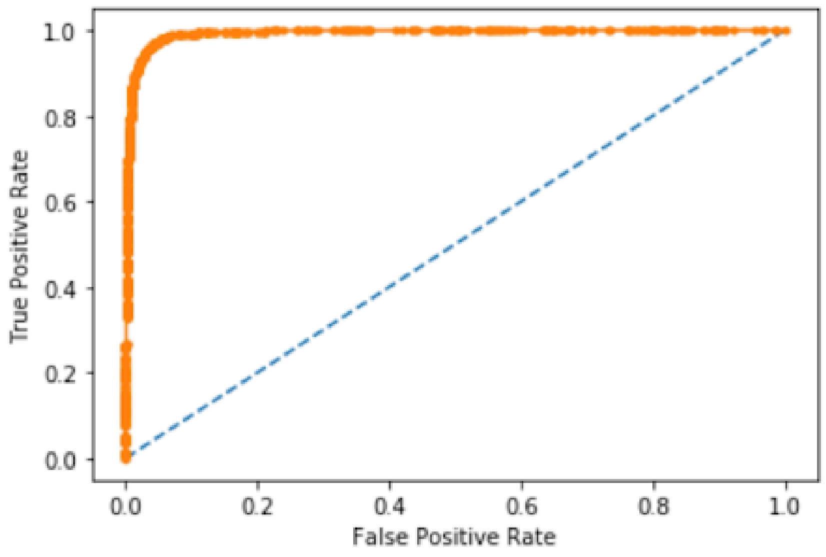


Figure 6.17: ROC of AdaBoost

For AdaBoost we get the AUC(Area Under Curve) score 0.993.

6.2.5 Support Vector Machines (SVM)

For Support Vector Machines (SVM) we get an accuracy score of 0.963.

The confusion matrix for Support Vector Machines (SVM) is as follows :

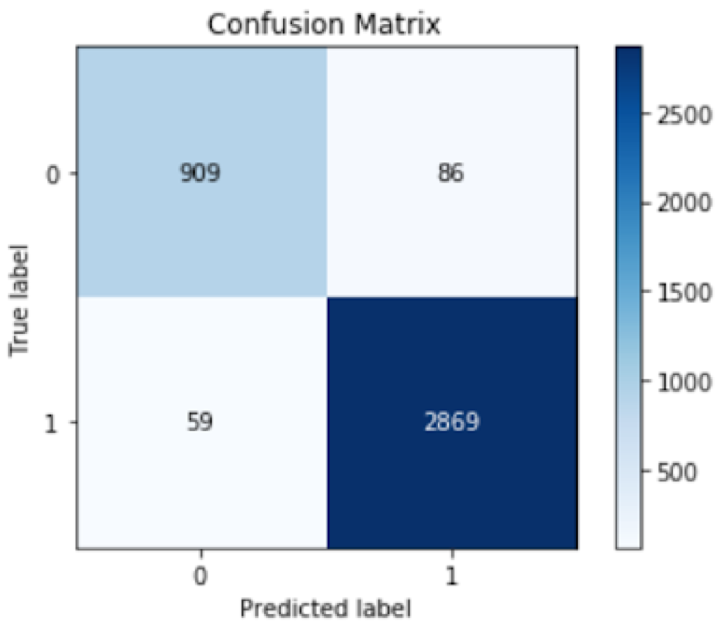
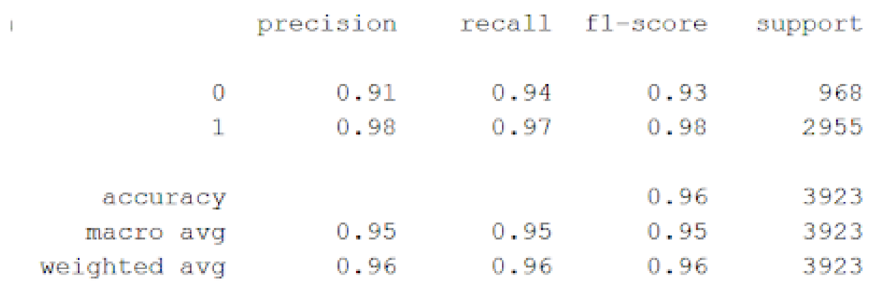


Figure 6.18: Classification Matrix of SVM

Here in the confusion matrix, the true negative value is nine hundred nine, the false positive value is eighty-six, the false-negative value is fifty-nine and the true positive value is two thousand eight hundred sixty-nine.

The classification report for Support Vector Machines (SVM)is as follows :



```
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       968
           1       0.98      0.97      0.98      2955

    accuracy                           0.96      3923
   macro avg       0.95      0.95      0.95      3923
weighted avg       0.96      0.96      0.96      3923
```

Figure 6.19: Classification Report of SVM

In the classification report of Support Vector Machines (SVM) , we can see that the precision, recall, and f1-score for benign is 0.91,0.94 and 0.93 respectively. For malware, the precision, recall, and f1-score are 0.98,0.97 and 0.98 respectively. Here the number of benign files detected is nine hundred sixty-eight and the number of malignant files detected is two thousand nine hundred fifty-five.

The roc curve for Support Vector Machines (SVM) is as follows :

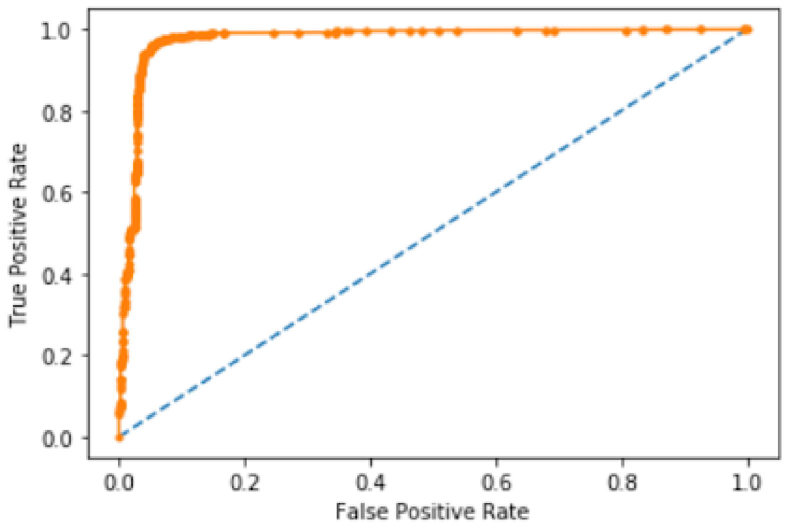


Figure 6.20: ROC of SVM

For Support Vector Machines (SVM) we get the AUC(Area Under Curve) score 0.976.

6.2.6 Decision Tree

For the Decision Tree, we get an accuracy score of 0.972.

The confusion matrix for Decision Tree is as follows :

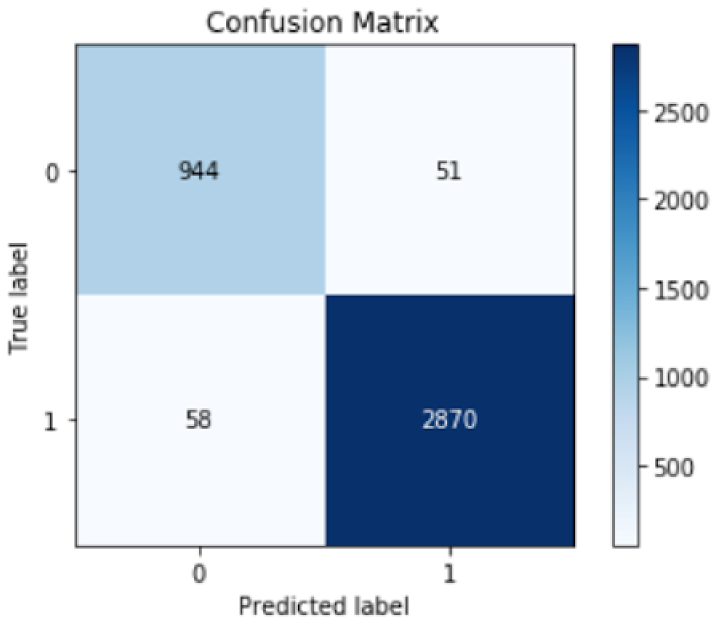
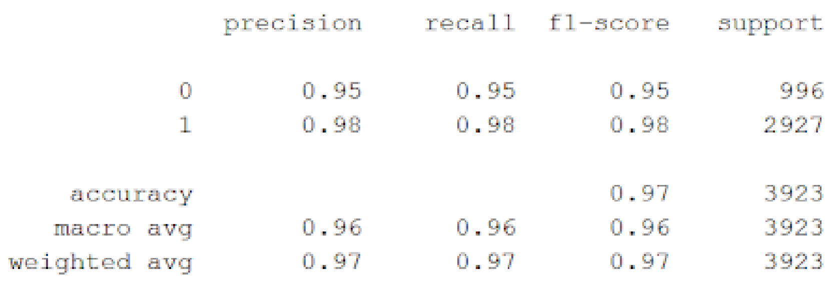


Figure 6.21: Classification Matrix of Decision Tree

Here in the confusion matrix, the true negative value is nine-hundred forty-four, the false positive value is fifty-one, the false-negative value is fifty-eight and the true positive value is two thousand eight hundred seventy.

The classification report for Decision Tree is as follows :



```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       996
           1       0.98      0.98      0.98      2927

    accuracy                           0.97      3923
   macro avg       0.96      0.96      0.96      3923
weighted avg       0.97      0.97      0.97      3923
```

Figure 6.22: Classification Report of Decision Tree

In the classification report of Decision Tree, we can see that the precision, recall, and f1-score for benign is 0.95,0.95 and 0.95 respectively. For malware, the precision, recall, and f1-score are 0.98,0.98 and 0.98 respectively. Here the number of benign files detected is nine hundred ninety-six and the number of malignant files detected is two thousand twenty-seven.

The roc curve for Decision Tree is as follows:

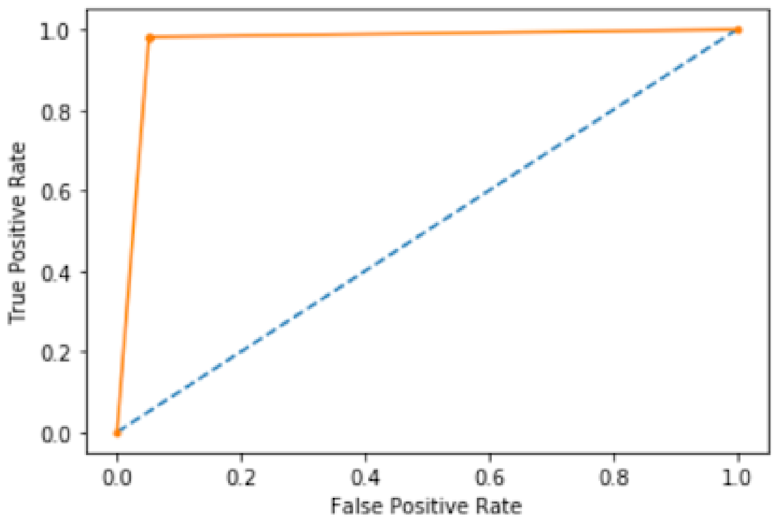


Figure 6.23: ROC of Decision Tree

For Decision Tree we get the AUC(Area Under Curve) score 0.964.

6.2.7 XGBoost

For XGBoost we get an accuracy score of 0.986.

The confusion matrix for XGBoost is as follow :

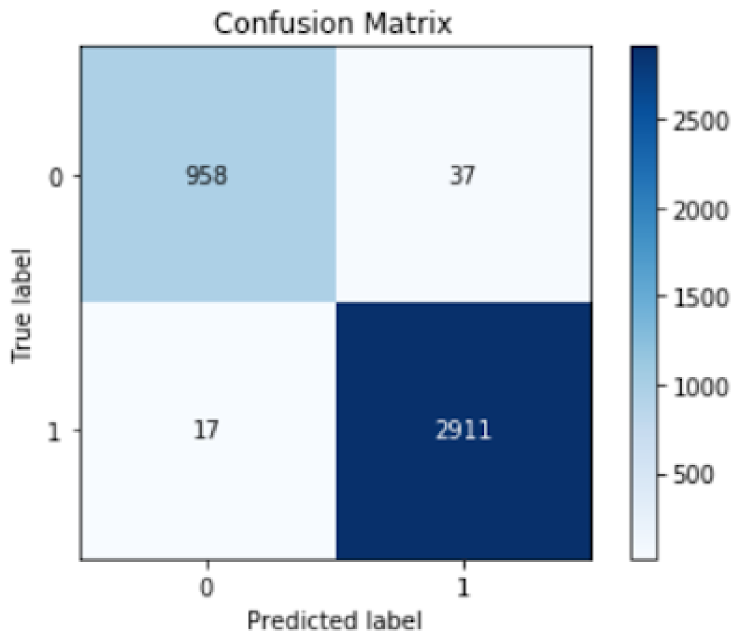
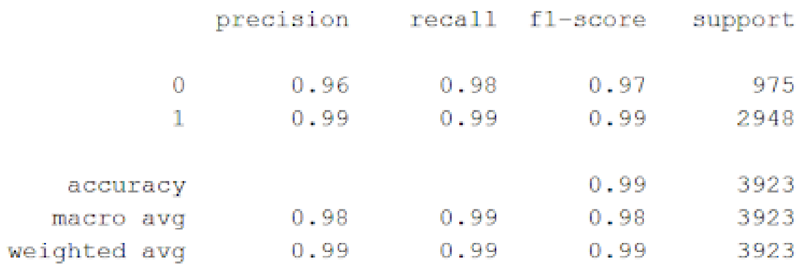


Figure 6.24: Classification Matrix of XGBoost

Here in the confusion matrix, the true negative value is nine hundred fifty-eight, the false positive value is thirty-seven, the false-negative value is seventeen and the true positive value is two thousand nine hundred and eleven.

The classification report for XGBoost is as follows :



```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       975
           1       0.99      0.99      0.99      2948

    accuracy                           0.99      3923
   macro avg       0.98      0.99      0.98      3923
weighted avg       0.99      0.99      0.99      3923
```

Figure 6.25: Classification Report of XGBoost

In the classification report of XGBoost, we can see that the precision, recall, and f1-score for benign is 0.96,0.98 and 0.97 respectively. For malware, the precision, recall, and f1-score are 0.99,0.99 and 0.99 respectively. Here the number of benign files detected is nine hundred seventy-five and the number of malignant files detected is two thousand forty-eight.

The roc curve for XGBoost is as follow :

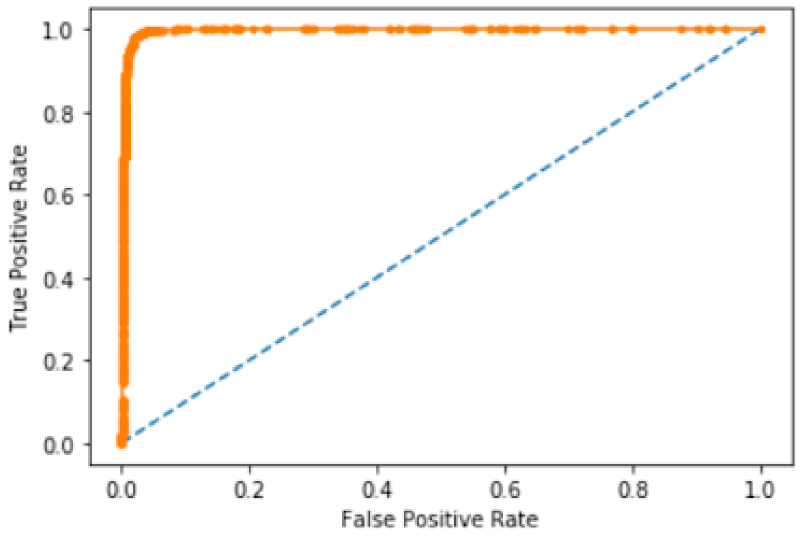


Figure 6.26: ROC of XGBoost

For XGBoost we get the AUC(Area Under Curve) score 0.996.

6.2.8 Artificial Neural Network (ANN)

For Artificial Neural Network (ANN) we get an accuracy score of 0.980.

The confusion matrix for Artificial Neural Network (ANN) is as follows :

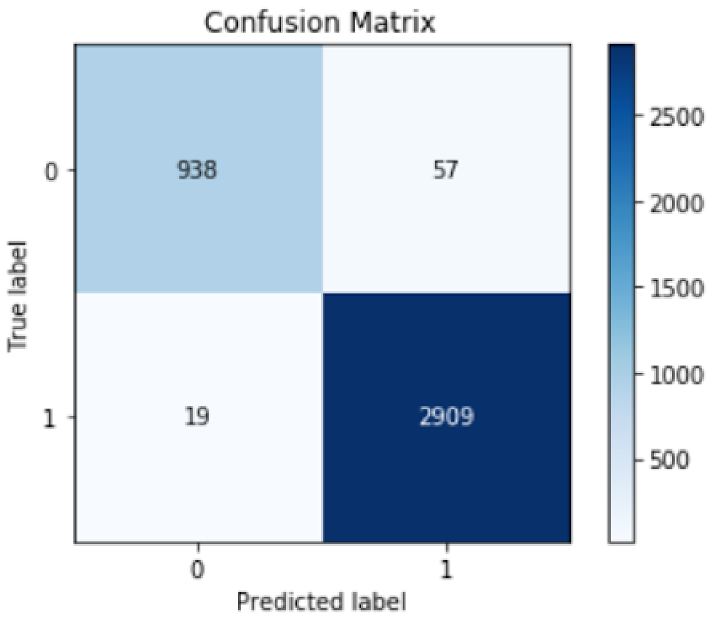
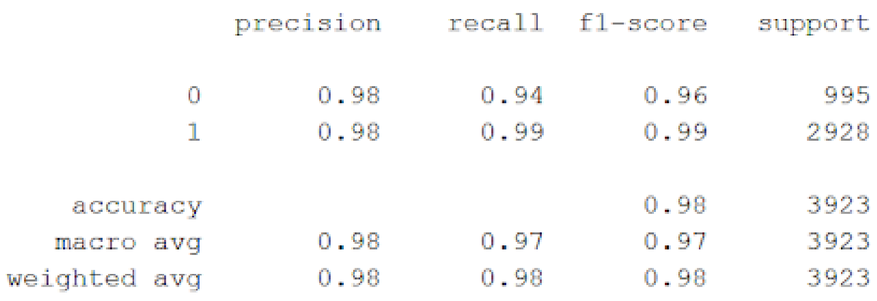


Figure 6.27: Classification Matrix of ANN

Here in the confusion matrix, the true negative value is nine hundred thirty-eight, the false positive value is fifty-seven, the false-negative value is nineteen and the true positive value is two thousand nine hundred nine.

The classification report for Artificial Neural Network (ANN) is as follows :



```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       995
           1       0.98      0.99      0.99      2928

    accuracy                           0.98      3923
   macro avg       0.98      0.97      0.97      3923
weighted avg       0.98      0.98      0.98      3923
```

Figure 6.28: Classification Report of ANN

In the classification report of Artificial Neural Network (ANN), we can see that the precision, recall, and f1-score for benign is 0.98,0.94 and 0.96 respectively. For malware, the precision, recall, and f1-score are 0.98,0.99 and 0.99 respectively. Here the number of benign files detected is nine hundred ninety-five and the number of malignant files detected is two thousand nine hundred twnty-eight.

The roc curve for Artificial Neural Network (ANN) is as follows :

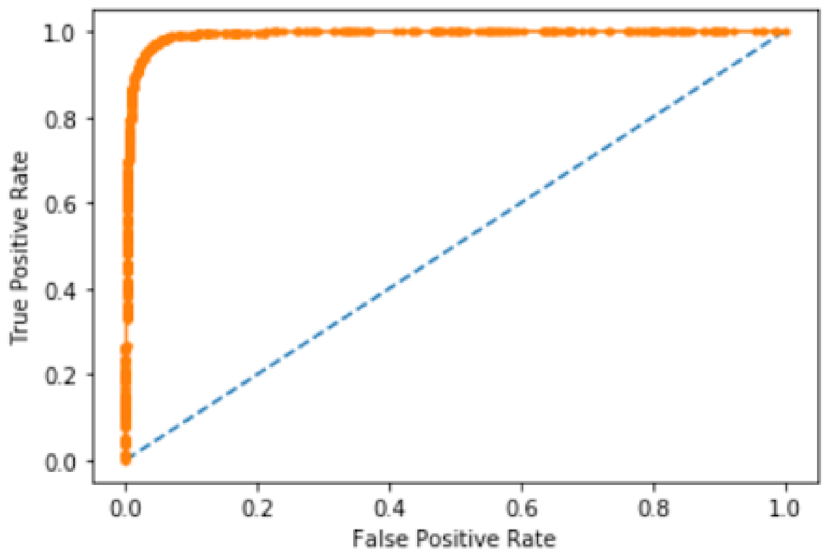


Figure 6.29: ROC of ANN

For Artificial Neural Network (ANN) we get the AUC(Area Under Curve) score 0.964.

6.2.9 Extra Tree Classifier

For Extra Tree Classifier we get an accuracy score of 0.959.

The confusion matrix for Extra Tree Classifier is as follows :

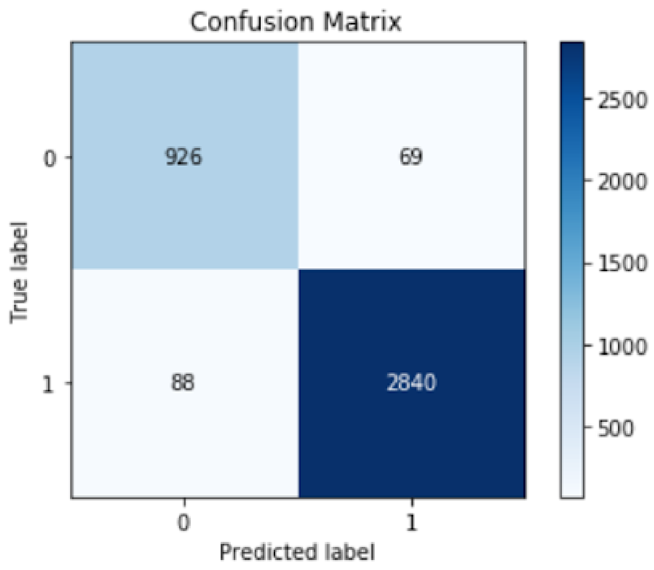
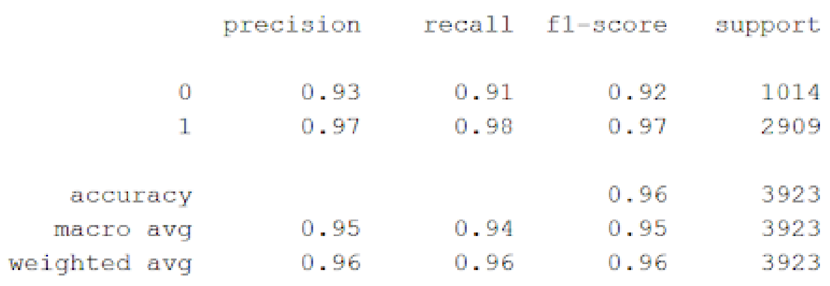


Figure 6.30: Classification Matrix of Extra Tree Classifier

Here in the confusion matrix, the true negative value is nine hundred twenty-six, the false positive value is sixty-nine, the false-negative value is eighty-eight and the true positive value is two thousand eight hundred forty.

The classification report for Extra Tree Classifier is as follows :



```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92      1014
           1       0.97      0.98      0.97      2909

    accuracy                           0.96      3923
   macro avg       0.95      0.94      0.95      3923
weighted avg       0.96      0.96      0.96      3923
```

Figure 6.31: Classification Report of Extra Tree Classifier

In the classification report of Extra Tree Classifier, we can see that the precision, recall, and f1-score for benign is 0.93,0.91 and 0.92 respectively. For malware, the precision, recall, and f1-score are 0.97,0.98 and 0.97 respectively. Here the number of benign files detected is one thousand fourteen and the number of malignant files detected is two thousand nine.

The roc curve for Extra Tree Classifier is as follow :

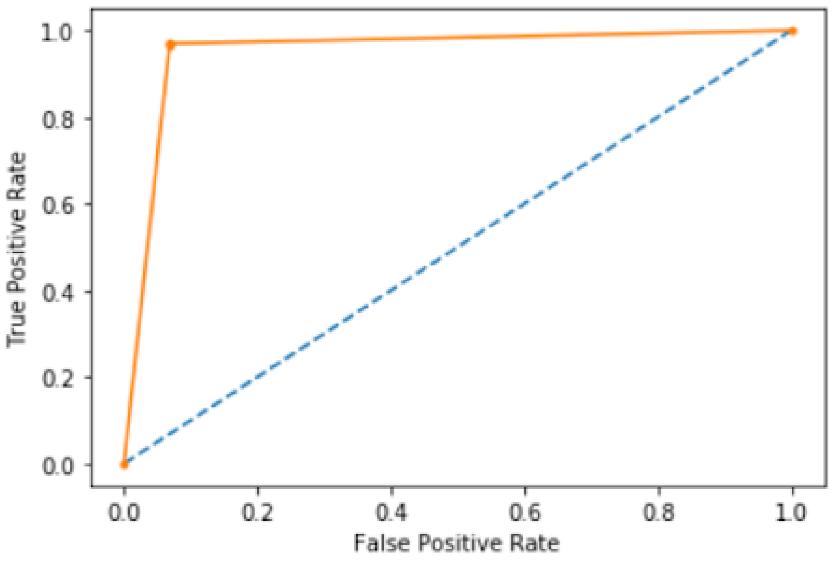


Figure 6.32: ROC of Extra Tree Classifier

For Extra Tree Classifier we get the AUC(Area Under Curve) score 0.952.

The result of the experiment is given below. It depicts that among all the classifiers XGBoost is performing best. For small to medium size, tabular data most of the time decision tree-based algorithm performs best. Since we are using tabular data, we used many decision tree-based algorithms. Among the decision tree-based algorithm XGBoost performs best due to its features like regularization, cross-validation, tree pruning, etc.

Table 6.2: Experimental Results

Algorithm	Accuracy	0			1			TPR	FPR
		Precision	Recall	f1-score	Precision	Recall	f1-score		
Logistic Regression	96.3	0.92	0.93	0.93	0.98	0.97	0.98	0.974	0.075
KNN	95.9	0.92	0.92	0.92	0.97	0.97	0.97	0.974	0.083
Random Forest	98.1	0.94	0.98	0.96	0.99	0.98	0.99	0.994	0.057
AdaBoost	97.2	0.94	0.95	0.94	0.98	0.98	0.98	0.984	0.064
SVM	96.3	0.91	0.94	0.93	0.98	0.97	0.98	0.979	0.077
Decision Tree	97.2	0.95	0.95	0.95	0.98	0.98	0.98	0.980	0.051
XGBoost	98.6	0.96	0.98	0.97	0.99	0.99	0.99	0.994	0.037
Artificial Neural Network	98.0	0.98	0.94	0.96	0.98	0.99	0.99	0.993	0.057
Extra Tree Classifier	95.9	0.93	0.91	0.92	0.97	0.98	0.97	0.969	0.069

The ROC Curve of all the classifier is given below:

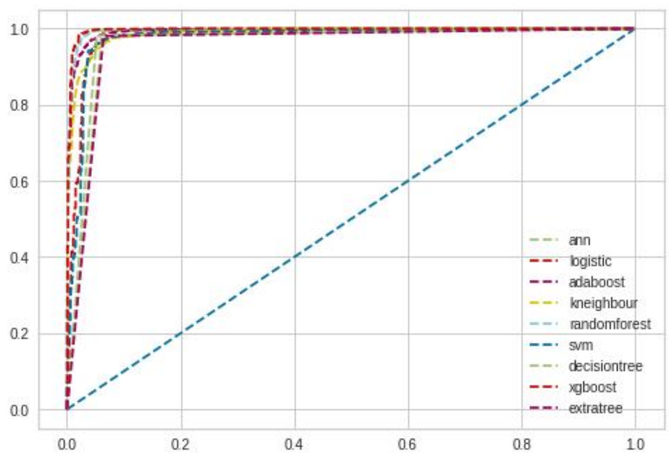


Figure 6.33: ROC Curve For All Classifiers

The accuracy rate comparison of different algorithms is given below:

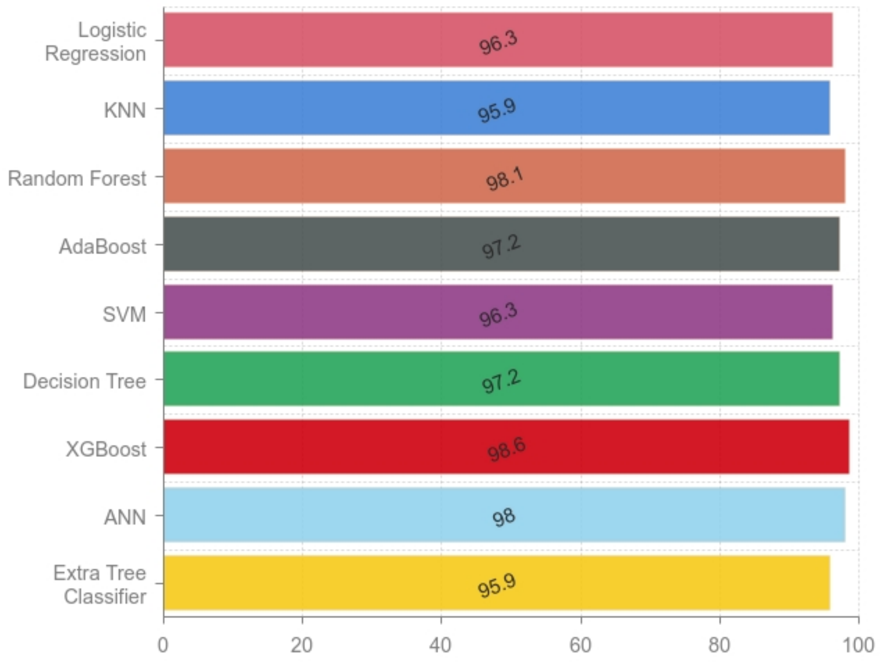


Figure 6.34: Accuracy Rate of all algorithms

The support score of different algorithms is given below:

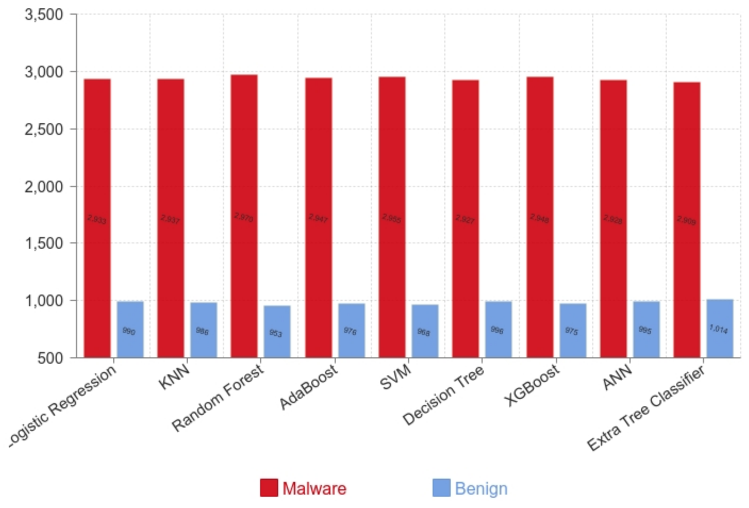
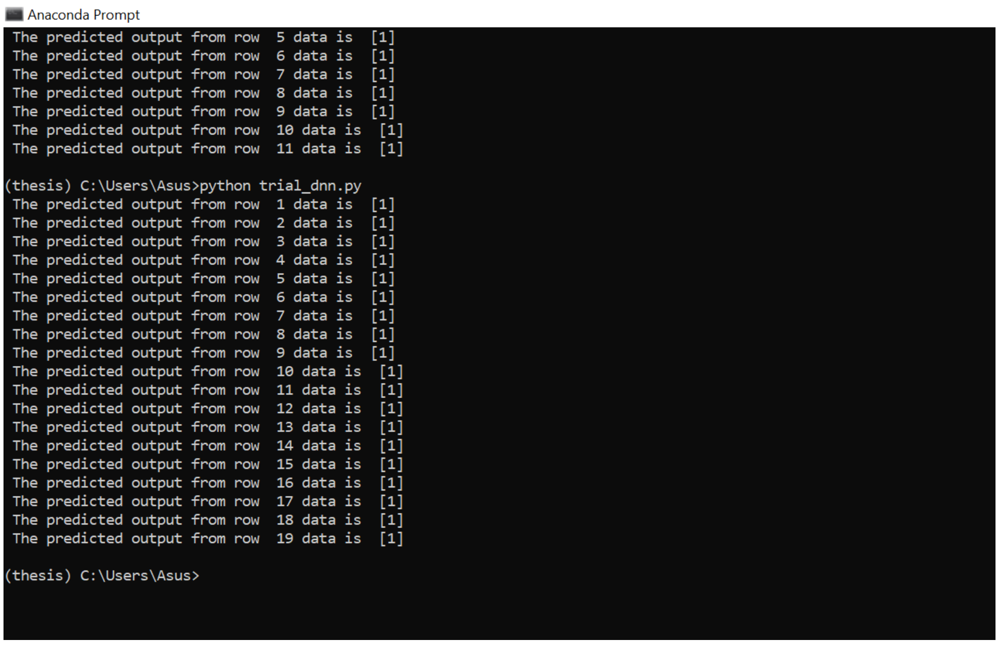


Figure 6.35: Support Score of all algorithms

6.3 Real-time analysis

We set up a server and a client with the help of the Python Flask. On the client-side, we implemented our machine learning classifier. The data got by the client is passed to the trained classifier, the trained classifier then predicts the label of the data. In our case, it took around 8 seconds for the classifier to classify the data in real-time. The following figure shows the output of the classifier on the client-side.



```
Anaconda Prompt
 The predicted output from row  5 data is  [1]
 The predicted output from row  6 data is  [1]
 The predicted output from row  7 data is  [1]
 The predicted output from row  8 data is  [1]
 The predicted output from row  9 data is  [1]
 The predicted output from row  10 data is  [1]
 The predicted output from row  11 data is  [1]

(thesis) C:\Users\Asus>python trial_dnn.py
 The predicted output from row  1 data is  [1]
 The predicted output from row  2 data is  [1]
 The predicted output from row  3 data is  [1]
 The predicted output from row  4 data is  [1]
 The predicted output from row  5 data is  [1]
 The predicted output from row  6 data is  [1]
 The predicted output from row  7 data is  [1]
 The predicted output from row  8 data is  [1]
 The predicted output from row  9 data is  [1]
 The predicted output from row  10 data is  [1]
 The predicted output from row  11 data is  [1]
 The predicted output from row  12 data is  [1]
 The predicted output from row  13 data is  [1]
 The predicted output from row  14 data is  [1]
 The predicted output from row  15 data is  [1]
 The predicted output from row  16 data is  [1]
 The predicted output from row  17 data is  [1]
 The predicted output from row  18 data is  [1]
 The predicted output from row  19 data is  [1]

(thesis) C:\Users\Asus>
```

Figure 6.36: Output from Flask client

Chapter 7

Conclusion and future work

In our work, we used multiple methods such as KNN, logistic regression, SVM, XG-Boost, Decision Tree etc classifiers are used along with the artificial neural network. The dataset used in the proposed model had 19611 samples and the dataset was labeled. All the nine methods are compared that were used to detect the malware. After comparing all the three methods, we found that XGBoost got the highest accuracy of 98.6 percent whereas TPR is 0.99 and FPR is 0.037 with an AUC of 0.99. The second-best accuracy we got from Random Forest was 98.1 percecnt.KNN and Extra Tree Classifier got the lowest accuracy which was 95.9 percent. We also built a client-server model by using Flask for real-time implementation to see if it could actually detect the unknown data as malware or benign. In any case, completely assess the reasonableness of our methodology, a lot more examination need to direct. While our underlying outcomes are promising, more works are needed to improve the technique and accuracy of our proposed model. For the future, we will consider a more advanced neural network alongside ANN. We are planning to build a hybrid model and an antivirus in the future, in which we are going to implement the hybrid machine learning algorithm. The antivirus would detect the PE file and predict whether they are malware or benign. We hope it would be able to tackle zero-day attacks, which the current antivirus is not able to.

Bibliography

[1] *Malware of the 1980s: Looking back at the brain virus and the morris worm — welivesecurity*, https://www.welivesecurity.com/2018/11/05/malware-1980s-brain-virus-morris-worm/, (Accessed on 12/21/2019).

[2] *What is the jerusalem virus? - definition from techopedia*, https://www.techopedia.com/definition/27875/jerusalem-virus, (Accessed on 12/21/2019).

[3] *The cih virus: What it is and how to remove it*, https://www.lifewire.com/what-is-the-cih-virus-4769409, (Accessed on 12/21/2019).

[4] *10 cyber security facts and statistics for 2018*, https://us.norton.com/internetsecurity-emerging-threats-10-facts-about-todays-cybersecurity-landscape-that-you-should-know.html, (Accessed on 12/06/2019).

[5] *How common is identity theft? (updated 2018) the latest stats*, https://www.lifelock.com/learn-identity-theft-resources-how-common-is-identity-theft.html, (Accessed on 12/06/2019).

[6] *Internet security threat report (istr) 2019*, https://www.symantec.com/en/in/security-center/threat-report.

[7] *Top 10 malware january 2019*, https://www.cisecurity.org/blog/top-10-malware-january-2019/, (Accessed on 12/06/2019).

[8] *Ransomware prevention and response for cisos — fbi*, https://www.fbi.gov/file-repository/ransomware-prevention-and-response-for-cisos.pdf/view, (Accessed on 12/06/2019).

[9] *Cybercrimes remain most worrisome to americans*, https://news.gallup.com/poll/244676/cybercrimes-remain-worrisome-americans.aspx, (Accessed on 12/06/2019).

[10] *Cybersecurity consulting services*, https://www.accenture.com/us-en/about/security-index, (Accessed on 12/06/2019).

[11] *A brief history of malware, its evolution and impact*, https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/, (Accessed on 12/06/2019).

[12] *Flashback friday: Sql slammer — welivesecurity*, https://www.welivesecurity.com/2016/09/30/flashback-friday-sql-slammer/, (Accessed on 12/07/2019).

[13] *How sircam started and its legacy*, https://www.exabeam.com/information-security/sircam/, (Accessed on 12/07/2019).

[14] *Conficker's estimated economic cost? $9.1 billion*, https://www.zdnet.com/article/confickers-estimated-economic-cost-9-1-billion/, (Accessed on 12/07/2019).

[15] *11 malware attacks that nearly wrecked the internet*, www.popularmechanics.com/technology/security/g29625471/history-of-malware-attacks/, (Accessed on 12/07/2019).

[16] *Ransomware everywhere — symantec connect community*, https://www.symantec.com/connect/articles/ransomware-everywhere, (Accessed on 12/07/2019).

[17] *January 2018 attack vectors – hackmageddon*, https://www.hackmageddon.com/2018/02/22/january-2018-cyber-attacks-statistics/january-2018-attack-vectors/, (Accessed on 12/07/2019).

[18] *What you must know about machine learning malware analysis - by*, https://hackernoon.com/what-you-must-know-about-machine-learning-malware-analysis, (Accessed on 12/07/2019).

[19] A. Kumar, K. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set", *Journal of King Saud University-Computer and Information Sciences*, 2017.

[20] P. Singhal and N. Raul, "Malware detection module using machine learning algorithms to assist in centralized security in enterprise networks", *arXiv preprint arXiv:1205.3062*, 2012.

[21] M. Belaoued and S. Mazouzi, "A real-time pe-malware detection system based on chi-square test and pe-file features", in *IFIP International Conference on Computer Science and its Applications*, Springer, 2015, pp. 416–425.

[22] M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, and C. Park, "Flow-based malware detection using convolutional neural network", in *2018 International Conference on Information Networking (ICOIN)*, IEEE, 2018, pp. 910–913.

[23] H. El Merabet and A. Hajraoui, "A survey of malware detection techniques based on machine learning", *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 10, no. 1, pp. 366–373, 2019.

[24] R. R. Yang, V. Kang, S. Albouq, and M. A. Zohdy, "Application of hybrid machine learning to detect and remove malware", *Transactions on Machine Learning and Artificial Intelligence*, vol. 3, no. 4, pp. 16–16, 2015.

[25] H. S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models", *arXiv preprint arXiv:1804.04637*, 2018.

[26] *What is noisy data? - definition from whatis.com*, searchbusinessanalytics.techtarget.com/definition/noisy-data, (Accessed on 12/07/2019).

[27] *Feature selection techniques in machine learning with python*, https://www.towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python, (Accessed on 12/07/2019).

[28] *Chapter 6: Model training with machine learning - data science primer*, https://elitedatascience.com/model-training, (Accessed on 12/07/2019).

[29] *Pe format - win32 apps — microsoft docs*, https://docs.microsoft.com/en-us/windows/win32/debug/pe-format, (Accessed on 12/25/2019).

[30] *Exploit development 02 — pe file format 1 - milad kahsari alhadi - medium*, https://medium.com/@MKahsari/exploit-development-02-pe-file-format-1-998c252b5670, (Accessed on 12/25/2019).

[31] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: A methodology review", *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[32] *How random forest algorithm works in machine learning*, https://syncedreview.com/2017/10/24/how-random-forest-algorithm-works-in-machine-learning/, (Accessed on 12/25/2019).

[33] *Understanding adaboost - towards data science*, https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe, (Accessed on 12/25/2019).

[34] T. Joachims, "Support vector and kernel methods", *SIGIR 2003 Tutorial*, 2003.

[35] M. Kruczkowski and E. N. Szynkiewicz, "Support vector machine for malware analysis and classification", in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*, IEEE Computer Society, 2014, pp. 415–420.

[36] *Decision tree introduction with example - geeksforgeeks*, www.geeksforgeeks.org/decision-tree-introduction-example/, (Accessed on 12/07/2019).

[37] S. Hutchinson and U. Karabiyik, "Forensic analysis of spy applications in android devices", 2019.