PE MALWARE DETECTION USING ENSEMBLE LEARNING

Sabari Raja

Applied project submitted in partial fulfilment of the requirements for the degree of M.Sc. in Data Analytics at Dublin Business School

Supervisor:

Shubham Sharma

May 2022

DECLARATION

I, Sabari Raja declare that this applied project that I have submitted to Dublin Business School for the award of Master of Science in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: **Sabari Raja**

Student Number: **10582377**

Date: **23rd May 2022**

ACKNOWLEDGEMENTS

I would like to take this opportunity and thank my supervisor, Ms. Shubham Sharma for her guidance and support throughout this journey to complete this applied project. I would also like to thank all my lecturers and academic staffs for their tremendous help, in making me reach this far in my master's applied project. This journey would have not been possible if it wasn't their support. I also want to express my gratitude to my course coordinator Dr. Andrew Browne who was very kind to help me whenever required. I am thankful to all my friends who helped me during data collection. Finally, I would like to thank my family and friends for support and constant source of inspiration and almighty God, for his grace in me.

ABSTRACT

In the ever rising age of the internet, where security breaches have become very common the safety of users have been put on threat. This is majorly observed to be done through evolving viruses. One such virus is known as a malware; which was initially detected in 1970 and has witnessed to grow exponentially. Therefore this creates a challenging task in the digital environment with respect to complexity and volume; leading to loss of information and replacement of malicious codes. Also, with existing traditional methods which are mostly time consuming and unreliable, it becomes quite easy for malware creators to get away with it. Hence, with this purpose, comes the need to develop intelligent and automatic detection techniques that could not only detect but also classify malwares as, good-ware and benign. Hence, I propose the implementation of Machine Learning based hybrid ensemble algorithms that could discover portable executable files by examining their static features. Furthermore, I modify our original model to enrich the extractable information, by implementing on a dataset whose content range from unambiguous malware and good-ware samples to more ambiguous and real ones. A total of four ML based algorithms, one stacking algorithm and two deep learning algorithms are trained and tested on this dataset, with stacking algorithms achieving the highest accuracy of 99.20 percent. Selecting the number of features was based on research of previous studies. This thesis confirms that it is possible to use machine learning for static malware detection. It can also help for future automated malware analysis research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In current history, the number of reported security breaches caused by viruses, Trojans, ransomwares, and other malware has risen sharply, with reports of malware infections making the news more than ever. A new technical issue is encountered almost every week, which could be perceived as the security community's failure to control and detect malicious content. According to ITU (International Telecommunication Union) statistics, by the end of 2020, two-thirds of the world's population will have connection to the web. All the users of the Internet are generally in a vulnerable state, due to multiple cybercriminals making use of hacking methodologies. These methods are used to majorly target and damage the normal working and execution of a programming service in a computer system. These cybercriminals utilizes various malicious software's that would result into exploitation of basic services and result into harming the user device. The malwares used by such hackers are specifically designed by software developers and is used with the intention to harm the computer system of a user and achieve personal gain out of it. Such malwares can further be classified into various categories depending on the action it intends to perform. A major chunk of common viruses include worms, adware, spyware and Trojan horses (Namanya, A.P.; Cullen, 2018). On the other hand, multiple illegal activities such as hacking and breaching have been constantly evolving into the criminal world of business. Therefore, the number of phishing attacks (Azeez, N.A.; Salaudeen, 2020) experienced by users has also been on rise and includes various intrusions that further involve higher complex instrumentation of already hacked machines. With increase in not only software related malware, there has also been a rise in development of non-human agents such as botnets. These botnets are meant to be controlled by utilizing the concepts of Internet of Things (IoT) which are further connected to multiple devices that would result into breaching of private information (Yong, 2020). Along with botnets, there has also been an observed increase in internet crime with regards to Wireless Sensor Networks (WSN's). However, it is still difficult to ignore the fact that the implementation and usage of malwares by hackers has possessed to be one of the most serious threats to online users including people from corporate and hospitals. These malware strategies are responsible to convert executable code into active content and safeguard it from any vulnerable attacks and prevent it from illegal breaching (Boukhtouta, A.; Mokhov, 2016). The implementation of a regular malware results into various forms of malfunctions in a computer system and also it does possess the potential to not only slow down the system but also crash a computer system when in use by the client. This slowing down of a computer system is further accompanied by reduction in disc and memory space, increased unnecessary internet activity, and arrival of unexpected pop-ups. On the other hand, such malwares are also used to gain access to personal information of the user such as his personal details, bank account details and login credentials (Odusami, Volume 942). In addition to this, ransomware is another form of malware that locks the user from their respective systems and forces them to pay a certain amount of money resulting in a decided ransom which is generally in the form of cryptocurrency; thereby making it untraceable to reach out to the real person. Hence, the user is victimized to pay the ransom to get back his access to the system.

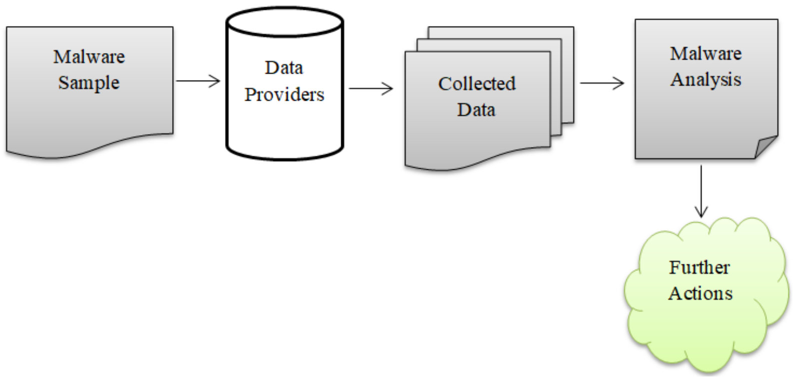


Figure 1: Workflow of a Malware

1.1 Background

Malware is software that is designed to harm users, networks, or computers. Malware's increasing speed, volume, and complexity pose a threat to everyone in the digital world, including businesses, governments, and individual users. It's critical to stay safe and have a strong intrusion barrier in today's digital environment because malware is one of the most common intrusion techniques used to launch an attack.

(Bazrafshan, Z.; Hashemi, 2013) looked at signature-based, behavior-based, and heuristic-based malware detection methods. The study did not, however, look into computational methods or dynamic or hybrid malware detection systems. A similar experiment was carried out by (Souri, A.; Hosseini, 2018) using two of the three mechanisms. Furthermore, knowledge discovery and machine learning initiatives for detecting and classifying malicious files were not considered in the study. (Ye.et.al, 2017) provided an overview of malware detection and analysis methods. The authors of the study further proposed two angles for investigation of the same that included feature extraction and classification based on certain features so extracted and selected. As per the results obtained through their experiments it was observed that malware detection using data mining methods gave higher accuracy than other existing methods for malware detection while reducing the number of false positives.

However, the overall efficiency of the system was majorly impacted by the usage and choice of classifiers and features to be used. The authors also put forward the implementation of class labels that could further result into improving the accuracy of benign and good ware files (Rathore et al, 2019) for detecting malware. They also used certain techniques to develop these models, such as resolving the issue of class imbalance, cross-validation, and so on. For malware classification, they used machine instruction (opcode) frequency as a feature vector in both supervised and unsupervised learning. Feature reduction methods such as a single layer auto-encoder and a 3-layer stacked auto encoder were used to reduce dimensionality. A deep neural network (DNN) and a random forest were then used to perform the recognition (RF). The RF algorithm outperformed the DNN models, according to their findings. Deep learning may not be effective for malware detection, according to these findings.

1.2 Problem Statement

The first pitfall is the lack of an agreed-upon definition of malware; not only does this result in different anti-virus vendors classifying the same program differently, but it also means that some programs fall into a grey area for which no clear classification can be deemed correct. Adware, or advertising-supported software, is an example of software that, while not directly malicious, performs arguably non-requested actions. The lack of concrete metrics and properties that distinguish malware from good-ware necessitates additional effort in the preparation of datasets for malware detector evaluation. Because of its complexity, detecting all malware is difficult. Current methods have flaws, and in order to keep up with the rate of change, automation is required. Machine learning techniques can be used to both automate and detect new malware as a solution.

It's not easy to use machine learning on malware, but it's been done for years. And, because malware is so complex, there would always be a competitive race between securing the data and breaching the data. As a result, research in this field will continue to be relevant. Even if researchers devise a new method for detecting all malware, malware authors will eventually devise a way to circumvent it. According to IBM Security, the average cost of a security breach is $4.98 million, and it takes an average of 280 days to detect and contain a breach. It is not only large corporations that can afford this cost, but the healthcare industry is the most targeted. Because there is so much information about malware classification, the work proposed in this thesis is necessary. In addition to this, these methods are highly optimized for that dataset. It is possible that a method that performs well in the lab will not perform as well in the real world. As a result, three cutting-edge malware detection methods are chosen to be compared on the same dataset, to see if they can perform well on a dataset that they were not designed to be trained on.

1.3 Motivation

Malware detection refers to the process of detecting malware on a network or on end devices (computers, telephones, etc.) Today, there are two approaches to malware detection: signature-based and non-signature-based. The detection of signature based technique majorly involves a sequence of bytes that are considered to be signatures and on the other hand, non-signature detection methods majorly involve a comparison to be done amongst existing patterns and behavior of malwares (the time between discovering vulnerability and updating or patching signatures) is equal to the sum of detection time, signature generation time, and updating time. Non-signature based detection, on the other hand, uncovers unknown malware such as zero-days, which are unintentional vulnerabilities that are unknown to the target software's vendor but known to the attacker. However, the non-signature-based approach is more prone to false positives, which means that even benign software can be classified as malicious.

1.4 Research Questions

Malware detection is a hot topic in research, with a slew of unsolved problems. Novel deep-learning frameworks must be proposed and validated on new malware datasets to address these issues. Previously, the use of ensemble methods such as random forests improved malware detection in user environments by enhancing machine learning models. In practice, the traditional signature-based approach to malware detection is ineffective at detecting unknown computer viruses. Antivirus programs must be updated on a regular and timely basis in order to maintain an appropriate level of server protection. Despite this, the time it takes antivirus companies to respond to the appearance of unknown malware can be several days. Erupting malware programs can cause irreversible system disruption during this time. Intelligent information protection systems can be developed thanks to such systems' ability to learn and generalize. The primary goal of this research report on malware detection is to determine whether the malware is malicious or benign. This is also the main goal of this project, because determining whether a block of code is harmful or not is the foundation of the entire detection algorithm. The main goal of this thesis is to create and test a machine learning-based neural network for analyzing portable executable files and classifying them as malicious or benign.

Following are the research question that needs to be explored for accurate predictive modelling of the system:

- What are the existing approaches to detect the same?
- To what extent is accuracy achieved in existing approaches?
- How ensemble modelling can enhance the overall performance of the system?
- How many algorithms would be required to achieve the same?
- What will be the evaluation parameters?

1.5 Research Aim and Objectives

The primary aim of the thesis is to detect the presence of malware in PE files and further classify it as good ware or benign. For this purpose, I put forward the usage and implementation of machine learning techniques to achieve the same. Following are the objectives of the work so submitted:

- To deploy a feature engineering and automated technique that would detect the presence of malware in server system
- To identify the most suitable machine learning algorithm that would give optimized results to achieve the same
- To recognize features that results good ware files to fall trap into malware files

1.6 Organization of the Thesis

Since Machine Learning is observed to be a significant method to detect malware; this report proposes "PE Malware Detection using Machine Learning" that consists of two stages: feature extraction and classification. This study puts forward an ensemble learning-based malware detection method wherein, the end-stage classification is done by machine learning algorithms such as: decision tree, random forest, gradient boosting, and AdaBoosting. The results of experiments are proposed to be executed on Windows Portable Executable (PE) malware dataset.

The rest of the report is organized as follows. Previous works, including an adequate criticism of existing methods and approaches are discussed in section two of literature survey. System design and implementation plan are briefed under methodology section followed by obtained results in chapter 4; whereas chapter 5 sums up the conclusion preceded by references.

2. LITERATURE REVIEW

The primary aim of the thesis is to develop a system model that could detect and classify evaluated PE files as malware or benign. For this purpose, I made a thorough literature survey on the existing work to understand the overall working of a malicious system. It has been observed through my survey that multiple research authors have contributed their study towards enhancing the system breaches with detection of such malwares. This section highlights research work based on machine learning for detection of PE files.

(M.G. Schultz, E. Eskin, 2001) dedicated his work into developing an automated framework that could detect malicious PE files by making use of static features such as a PE Header and a String. To implement this successfully, he made use of a malware dataset and divided that into further two subsets that were majorly categorized as training and testing. In the training phase, the process of classification took place followed by developing a model that could allocate malware and good-ware files from the dataset. Whereas in the testing phase classifiers were used to calculate unknown binaries. In the later stages the author proposed to execute the model using three machine learning based algorithms such as NB, multiple NB, and Ripper. He performed these algorithms on a dataset containing a total of 4266 samples wherein 1000 amongst these were considered to be benign. The author accomplished a total accuracy of 97.1% with his proposed system. In later researchers it was witnessed that the achieved accuracy was relatively higher when it was compared against existing techniques; such as signature based detection systems.

In a similar work by (B. Anderson, 2011) the author detected malwares based on a graph related analysis technique that worked on the concepts of raw binaries and opcodes. This approach was called as the n-gram approach wherein all the dynamic features were used for analysis. This generalized analysis majorly included control graphs and system calls. The author implemented this concept of n-gram approach and trained his dataset 776 benign samples and 780 malicious samples. He also proposed to implement machine learning based algorithm such as SVM and utilized kernel learning approach to find similarity index between graph edges. Authors such as (M. Eskandari, 2013) made use of dynamic features of a PE file to detect malwares in a system model. The dynamic features included API and system calls. In the later stages, he used machine learning based algorithms such as Bayesian classification to categorize benign and malware infected PE files.

In another research work by (P. Khodamoradi, 2015) the author developed a heuristic based detection model that was based on the metamorphic encryption technique and made use of static features for analysis. These static features included opcodes and API files that were further disassembled using IDA pro. In the next stage, all the static features were extracted using the same opcode. To accomplish a higher level of accuracy, the author proposed to implement the same concept by making use of six machine learning based algorithms including random forest and NB. However, in his research the author concluded that the overall enhancement of the system model was directly dependent on the classification algorithms that were used to detect the malware in a system. This model obtained an inclined accuracy with higher percentages in precision and recall factors.

(C. LeDoux, 2015) presented a summarized review on all the existing methods to detect malwares. In his review he also mentioned certain prevention techniques that could be adopted in the early stages. His work was majorly contributed to highlight the importance of machine learning based algorithms to detect PE based malware files.

In another study by (Liang, J. Pang, 2016) he studied the working nature of malwares based on behavior and how they performed on static and dynamic features. On one hand where static features included opcodes, there dynamic features included PE files from API calls and registry. All the mentioned features were pondered upon and examined accordingly. In the later stages, the author applied the concepts of supervised machine learning algorithms to classify the detected malwares as benign or good ware. He also proposed to detect multiple variants of the virus using Jaccard's similarity distance.

(D. Ucci, 2018) proposed his work by analyzing static analysis to extract sequences in the headers of a PE file. The author implemented this on a dataset containing 4783 samples and he was successfully able to categorize them on the basis of classification algorithms. He made use of random forest as the classification algorithm and proved to generate an accuracy of 96%. However, the work implemented in this model was later extended as a study in multiple research areas and was witnessed to be faster in terms of comparative computation to other detection systems.

In another work by (E. Gandotra, 2016) the author proposed a different detection method that could classify multiple variants of the malware. He proposed to achieve this by predicting signatures. However, his study highlighted to analyze static features of a PE file that included extracting strings. N-grams and API calls. In the later stages, the author divided the dataset into training, testing and cross validation phases. For the training stage, he implemented the concepts of unsupervised learning based machine learning algorithms and tested the system model against specific hyper parameters.

In 2017 authors (A. Damodaran, F. Di Troia, 2017) implemented the Markov model so that they could compare the obtained results of detection and classification based on dynamic and static analysis. The author proposed the execution of this model on different variants of the malware family. However, his study provided an extensive research area for implementation of detection approaches that were also based on data mining techniques. His study also included examining various feature set dimensions by using classification and clustering algorithms.

A similar work was observed by (Q.K.A. Mirza, 2018) wherein the author proposed solutions for malware detection using hybrid methods. In the later stages, data mining algorithms were used to classify malwares as benign and good ware.

(W. Wang, 2019) Evaluated three types of malware detection methods: signature-based, behavior-based, and heuristic-based. However the study did not examine computational methods, as well as dynamic and hybrid malware detection systems. (H. Zhang, 2019) carried out a similar experiment using two of the three mechanisms. Furthermore, the study did not consider knowledge discovery or machine learning initiatives for detecting and classifying malicious files.

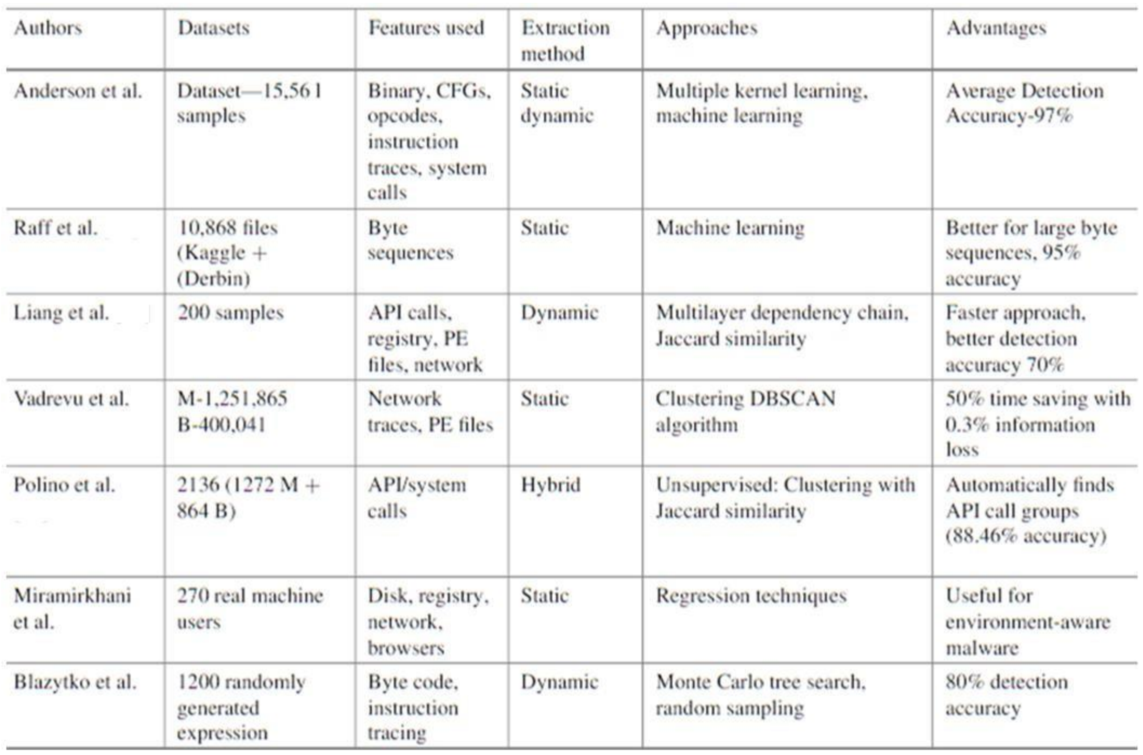
(M. Younas Syst., 2018) presented an overview of malware detection and analysis methods. They researched detection methods using specific sources from two aspects: feature extraction and grouping or classification. According to the report's results, data mining-based malware analysis structures could be used to obtain higher exactness in malware detection while lowering the number of false positives. The utilization of malware investigative techniques was strongly affected not only by the various classifiers used, but also by the features extracted, according to their observations. They also tried to suggest that instead of just class labels, a gathering of classification models could improve detection accuracy, and that an equitable distribution of harmful and benign files is required for training.

(Z. Ma, 2019) Malware detection was investigated using a variety of machine learning algorithms and deep learning models. They likewise utilized certain technology to develop these models, such as fixing the dispute of class imbalance, cross-validation, and so on. For malware analysis, they used machine instruction likelihood as a feature vector in both supervised and unsupervised learning. Dimension reduction methodologies such as a single -layered auto-encoder and a 3-layer stacked auto encoder have been used to reduce the number of features. A deep neural network (DNN) and a random forest were then used to perform the recognition (RF). The RF algorithm found to outperform the DNN models, according to their research results. Machine learning may not be effective for malware detection, according to these findings.

(Ren, Z; Wu Ad Hoc, 2021) suggested two deep neural networks, DexCNN and DexCRNN, to distinguish between benign and malicious Android app packages (APKs). DexCNN and DexCRNN accomplished 93.4 percent and 95.8 percent accuracy rate, respectively, in the experimentations.

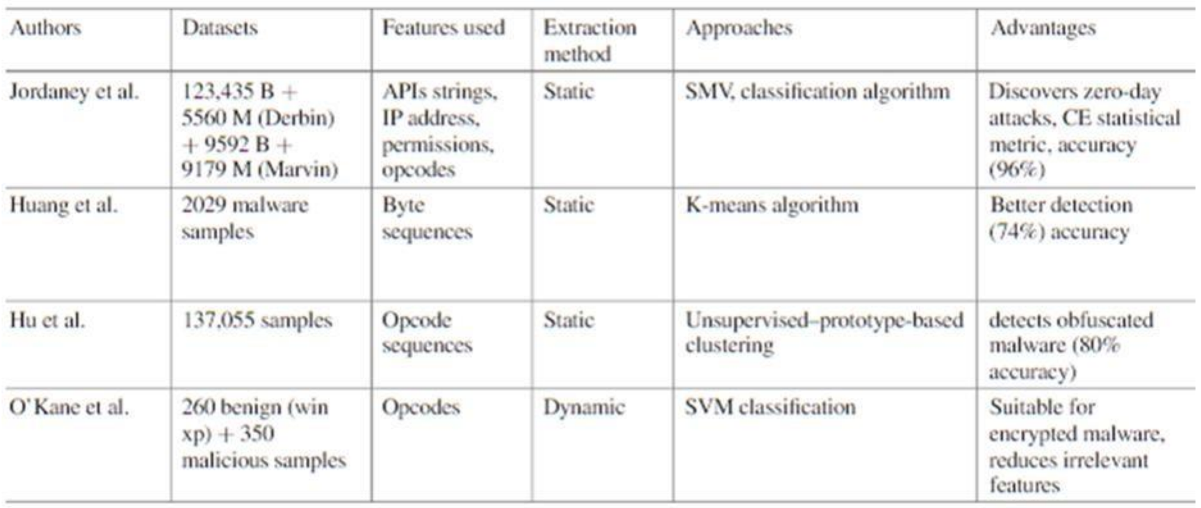
DBNs were used as an auto encoder by (Yuxin, 2019) to extract features from executable files. They evaluated the efficiency of DBNs as different classifiers to baseline malware detection models (SVM, decision trees, and the k-nearest neighbours algorithm), finding that the DBN significantly outperform the baseline models significantly. (Pei, 2020) proposed an improved model for Android malware detection that included graph convolutional networks (GCNs) for learning conceptual and measurable trends and an autonomously recurrent neural network (IndRNN) for understanding deep semantic features and extracting perspective features for ransomware recognition. (Ceponis and Goranin Computer, 2011) proposed using dual-flow deep neural networks for botnet recognition, such as a long short-term memory fully convolutional network (LSTM- FCN) and a gated recurrent unit (GRU)-FCN, and conducted experiments on the Windows OS calls traces dataset (AWSCTD), but found that conventional one-dimension single flow CNN produced the best performance.

Table 1: Literature Survey



Authors	Datasets	Features used	Extraction method	Approaches	Advantages
Anderson et al.	Dataset—15,561 samples	Binary, CFGs, opcodes, instruction traces, system calls	Static dynamic	Multiple kernel learning, machine learning	Average Detection Accuracy-97%
Raff et al.	10,868 files (Kaggle + (Derbin)	Byte sequences	Static	Machine learning	Better for large byte sequences, 95% accuracy
Liang et al.	200 samples	API calls, registry, PE files, network	Dynamic	Multilayer dependency chain, Jaccard similarity	Faster approach, better detection accuracy 70%
Vadrevu et al.	M-1,251,865 B-400,041	Network traces, PE files	Static	Clustering DBSCAN algorithm	50% time saving with 0.3% information loss
Polino et al.	2136 (1272 M + 864 B)	API/system calls	Hybrid	Unsupervised: Clustering with Jaccard similarity	Automatically finds API call groups (88.46% accuracy)
Miramirkhani et al.	270 real machine users	Disk, registry, network, browsers	Static	Regression techniques	Useful for environment-aware malware
Blazytko et al.	1200 randomly generated expression	Byte code, instruction tracing	Dynamic	Monte Carlo tree search, random sampling	80% detection accuracy

Table 2: Literature Survey



Authors	Datasets	Features used	Extraction method	Approaches	Advantages
Jordaney et al.	123,435 B + 5560 M (Derbin) + 9592 B + 9179 M (Marvin)	APIs strings, IP address, permissions, opcodes	Static	SMV, classification algorithm	Discovers zero-day attacks, CE statistical metric, accuracy (96%)
Huang et al.	2029 malware samples	Byte sequences	Static	K-means algorithm	Better detection (74%) accuracy
Hu et al.	137,055 samples	Opcode sequences	Static	Unsupervised–prototype-based clustering	detects obfuscated malware (80% accuracy)
O'Kane et al.	260 benign (win xp) + 350 malicious samples	Opcodes	Dynamic	SVM classification	Suitable for encrypted malware, reduces irrelevant features

2.1 Static Analysis

Static analysis examines malware samples without executing the code. When a quick decision must be made in a resource-efficient manner, static analysis is preferred. The comprehensiveness of the analysis and the parameters chosen for analysis determine the effectiveness of static analysis.

Source code, hashes, header information, strings, metadata, and other static data are all subject to static analysis. Because it may miss important malware functionality, simple static analysis is largely ineffective against sophisticated malware. To understand malware code, advanced static techniques use disassembler tools like IDA Pro, OllyDbg, and Olly Dump to reverse engineer it. Other tools, such as memory dumper LordPE, can help you get information about changes in the system's memory.

Features such as binary code, opcodes, strings, byte n-grams, and control flow graphs are used to further analyze these patterns. In signature-based detection methods, static analysis is used. Static analysis-based approaches are simple and quick, but they can't detect ever-evolving obfuscated malware correctly because some functions are left unidentified. Static analysis limitations have been investigated. These drawbacks give a dynamic analysis-based approach an advantage.

2.2 Dynamic Analysis

The malware samples are executed and analyzed in a real or controlled environment in dynamic analysis. A variety of debuggers can be used to perform dynamic analysis. API, system calls, instruction traces, registry changes, file writes, memory writes, and network changes can all be retrieved using other tools like Process Monitor, Regshot, Filemon, and Process Explorer.

The most common application of dynamic analysis is to comprehend the functionalities of malware samples under consideration. When environment aware malware detects that it is being executed in a controlled environment, it does not display its true behavior. As a result, dynamic analysis methods must include some real-world characteristics to make it difficult for malware to distinguish between the real and controlled environments.

Both analysis techniques are useful in different situations, but static analysis is preferred over dynamic analysis when the goal is to complete the analysis quickly while using the fewest resources possible.

Taking the above fact into account, some researchers proposed a new method called hybrid analysis, which combines the features of static and dynamic analysis. The authors used dynamic analysis to extract features for training and static analysis to test the detection system's efficiency.

We can use machine learning algorithms to create an automated malware analysis and detection system that can accurately distinguish between benign and malicious malware. These methods significantly improve detection efficiency while also reducing time and resource consumption.

These intelligent systems can also detect new malware quickly. As a result, a large number of authors have used machine learning algorithms to train and test models created using a large number of benign and malicious samples extracted using various static and dynamic analysis techniques.

The purpose of this thesis is to review and discuss malware analysis of PE files. PE files were chosen for this study because they work with Windows operating systems, and Windows is currently the most widely used operating system by users all over the world. PE is a 32/64-bit file format for executables, object codes, DLLs, and other Windows OS files.

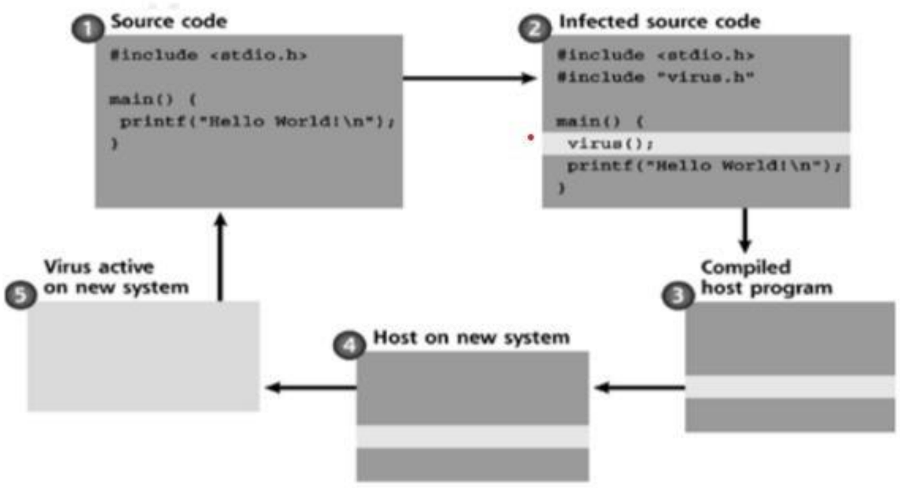
Malware analysis of PE files can include byte sequences, strings, information flow tracking, opcodes, control flow graphs, and API calls, among other things.

2.3 Malware in Computer Systems

With an advent in computer aided technology, release of viruses in a system has recently witnessed to be as a major digital threat to the world. Multiple forms and categories of malwares and viruses have been under observation and are increasing rapidly. With the necessity to satisfy and assure the safety of a hostile server, the entire computer system is being put at risk and large number of host have been assaulted. Apart from the increasing range of malwares, there is a relative decrease of the expertise level required in this field to improve the overall situation of the software market. Hence, securing such frameworks has become one of the most significant factors to protect online data so that the information of clients and organizations are reserved and not compromised upon.

Malware is considered to be as one of the most destructive forms of a virus code that is injected in a computer system with the sole purpose and intention of acquiring information of the user in an illegal form. Its representation is very much similar to that of normal software and hence is difficult to get recognized by naïve individuals. The intention of the intruder is to harm the data and release denial of service attacks, steal information in an illegal manner and question the integrity and confidentiality of the system. In order to successfully accomplish its goals, a malware is capable to perform various actions and operations in the form of code. It is capable to perform activities such as opening a file and reading a file through commands such as open, read, delete, modify, alter etc. It also possesses the capability to carry out registration activations and service activities. It can also start a program and end a program by giving certain commands. It can show a high impact on networking activities by connecting to servers through TCP, UDP, DNS and HTTP. Therefore, it has been observed that a malware can range from simple categories to that of complex forms wherein it becomes difficult for a legit individual to differentiate between a malicious user and a confidential mail coming from genuine source. However, one of the most important characteristic of a malware is that it can be controlled locally and also through the internet, making it highly portable and remote in nature of its execution.

Malware generally belongs to a larger family of illegal software's including spyware, Trojan horse, bots, worms etc. the figure below depicts the illustration of a virus infection occurring in software:



```
#include <stdio.h>

main() {
 printf("Hello World!\n");
}
```

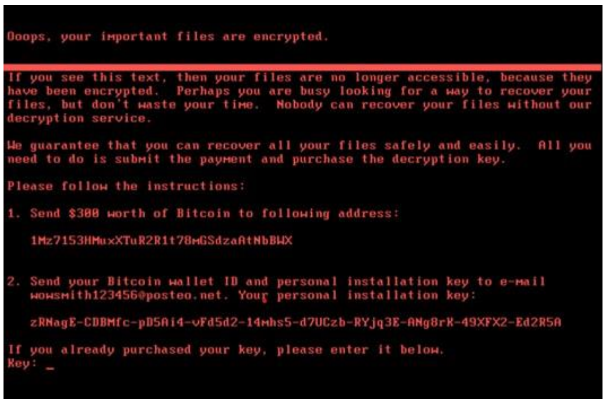
```
#include <stdio.h>
#include "virus.h"

main() {
 virus();
 printf("Hello World!\n");
}
```

Figure 2: Virus code (Emmanuel Masabo, 2018)

The different types of malwares are as follows:

- Ransomware: this malware is very specific with its execution and thereby intends to lock the PC screen with information that is majorly misleading and is used to cause concern to the legit user by making him to pay the ransom fee in order to regain access to the computer system.



```
Ooops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because they
have been encrypted.  Perhaps you are busy looking for a way to recover your
files, but don't waste your time.  Nobody can recover your files without our
decryption service.

We guarantee that you can recover all your files safely and easily.  All you
need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1. Send $300 worth of Bitcoin to following address:

   1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX

2. Send your Bitcoin wallet ID and personal installation key to e-mail
   wowsmith123456@posteo.net. Your personal installation key:

   zRNagE-CDBMfc-pD5Ai4-vFd5d2-14mhs5-d7UCzb-RYjq3E-ANg8rK-49XFX2-Ed2R5A

If you already purchased your key, please enter it below.
Key: _
```

Figure 3: Example of a Ransomware letter (vox.com, 2017)

- Virus: this malware possess the capability to internally hide within a program, replicate itself into multiple copies. However, it needs to pondered here that these multiple copies can further spread to other files and programs within a computer system.
- Trojan horse: this malware is able to disguise itself into some other program and tends to persuade the mind of the user by forcing him to install particular software which it offers. On installation, it carries out massive destruction within the computer system and increase the overall payload of the system.
- Rootkit: this malware tends to execute by eluding its presence and becoming invisible on the system. Thereby it executes its malfunctioning process by hiding in the server system and running viruses through it.
- Backdoor: this malware inhibits the capability to bypass all the normal authentication mechanisms and thereby leading to weakening the entire system by slowing down the networking connection of the system. This also leads to creation of many processes that are overall hidden and can be accessed in the future.
- Adware: this malware is responsible to develop and create multiple advertisements that are popped to the user when he tries to install or download any necessary software. The primary purpose of this malware is to generate income on behalf of the intruder who has created it.
- Exploit: the primary aim of this malicious code is to construct massive destruction in the computer system by targeting the vulnerabilities of the server that is to be exploited.
- Key loggers: this kind of a malware attack is responsible to record keyboard strokes and predicts its future pattern so that the intruder could steal useful and confidential information such as password credential and login details. This stolen information is

then further passed onto multiple authors who tend to produce similar form of attacking software codes.

- Potentially unwanted program: this malicious code is all about programs occurring on the server system of the user and tends to predict the files which are unwanted in the user's computer system. This leads to creation of illegitimate codes of virus leading to exposure of user's information and private data.
- Worm: this malicious code has a complex design structure that looks similar to that of a worm and is considered to be a sub-category of virus. However, it possesses the capability to spread in multiple devices without the help of external resource.
- Advanced persistence threats (APT): this form of a malware functions in a sophisticated manner and tends to exploit the overall vulnerabilities of a system. However, this malware can also be handled remotely and possess the capability to strike a specific target.
- Crypto Malware: this malware is also known as the data locker and is hence used to expose the access of data from the targeted system. The implementation of this malware usually involves the encryption process and abides the users from accessing their data. Hence the user is then liable to pay for encryption key, therefore in this way monetary benefits are generated for the authors who design such form of malwares.
- Locker malware: the primary aim of this kind of malware is to design a lock and force it to prevent the user's access from the server's interface, thereby resulting a large amount of data to be untouched in nature.
- Zeus: the goal of this malware is to steal banking information illegally from legit users by observing there keystroke patterns and logging in to their data files containing information of the customer such as his account details.

- Shadow-brokers: this malicious code is primarily responsible to exploit the overall vulnerability of the system and hence can be controlled remotely from any geographic location. These malicious codes also make use of webcams and microphones to gain access to user output and monitors customer keystrokes to access data illegally.

2.4 Malware Analysis Tools

2.4.1 Static Analysis Tools

- PEiD: this malware analysis tool is used to detect the type and category of a packer that is used to compile and build a program for a desired application. This toll can majorly help to identify certain specific programs that are required by the binary pack and is further responsible to calculate respective entropy of the system.
- UPX: this toll is an alternative to PEiD and is further used at the time of execution in compressing respective files. However, the process of packing makes any hardware tool available to analyse it and further detect it. But in later stages, these tend to shrink, hence UPX are used and it implements by hiding inside the original data and creating sub-packets of the system through operating systems.

2.4.2 Dynamic Analysis Tools

- Process Monitor: this analysis tool is used to control and monitor the real time of the networking system and is used to visualize all the middle events taking place.
- Dependency Walker: this tool is used to explore DLL based libraries of server system and is further used to export functions occurring between multiple layers of a DLL.
- Regshot: this toll is employed to capture screenshots once the malware is executed and thereby maintains a log of registry to keep the record of all the transactions taking place in an event.

2.5 Discussions

Since a thorough literature survey has been done in the previous section, it has been quite observed that research scholars have performed a massive scrutiny to not only detect the presence of virus, but also predict its occurrence in the future. This has led all the scholars to implement and examine techniques, concepts and methodologies of their own. Hence, it can be very well said that all the approaches in detecting and achieving the same motto differs and vary from author to author. With some authors such as (B. Anderson, 2011) implementing the techniques of machine learning, there have been observed authors who contributed their work in deep learning as well. Apart from the algorithms used by authors in the conducted literature survey, the perspective of selecting feature extraction and feature selection techniques have also differed to a great extent. In addition to this, it is worthy to note that the highest accuracy achieving algorithm was proven to be random forest and SVM in detecting the virus using machine learning whereas the implementation of CNN proved to generate optimized accuracy and minimized error loss while using the deep learning concept.

One of the keen observations made in the literature review is that, not many research scholars have presented their view with evaluation to malware aspects in their work, but have rather focused on detecting them. (A. Damodaran, F. Di Troia, 2017) contributed his work by implementing feature selection technique as an approach to predict benign files. The authors mainly comprehended the fact, of how malware differed from good ware and mentioned how its respective features should be selected. However, it can be said that through the literature survey so presented, a good framework can be understood to build an approach that would predict and classify PE malware files.

While comparing the existing work with the proposed thesis, a major factor that could be considered is the implementation of machine learning algorithms. As many authors have combined multiple algorithms to achieve a common goal, the same has been implemented in the thesis. Machine learning algorithms such as decision tress, random forest and logistic regression have been used; whereas deep learning algorithms such as CNN and LSTM has been used. Another comparative factor which seems to be common in existing and proposed work are the evaluation parameters. All the outputs obtained from the algorithms have been evaluated using evaluation metrics such as ROC curves, accuracy, and recall factor and confusion matrix.

On the other hand the contrasting factor which makes the proposed thesis different from the existing ones is the implementation of two concepts in one thesis. The concepts used were based on implementation of machine learning and deep learning. Apart from this, the second contrasting factor is the implementation of stacking algorithm or hybrid algorithm, which is also known as ensemble learning. The proposed thesis includes experimentation of stacking algorithm with random forest and Adaboost as Meta classifiers and decision trees as base the classifier.

Therefore, with the discussion so presented clarity of compare and contrast within existing and proposed work is successfully achieved.

3. METHODOLOGY USED

The concept of machine learning follows the primary study of computer aided algorithms that possess the capability to create a model based on previous samples, so that the model can take decisions and predict desired outcomes. This leads to development of a system that result into producing an optimized result and thereby enhancing the overall efficiency of the system. In such a scenario, the implementation of specific algorithms for certain applications play an important role. Therefore, it is necessary to select machine learning based algorithms that could predict and classify the desired application in an effective manner. Apart from algorithms, the methodologies used to implement the same also play a major role in deciding the evaluation parameters of the model so that the system could achieve maximum accuracy with less number of errors. This section of the thesis gives a detailed summary on the methodology involved to predict such applications based on machine learning.

3.1 Workflow of the Proposed System

The primary aim of any intruder is to attack the computer system and inject it with malicious codes that would result into breaching of users personal data. However, traditional methods used by detecting clients have succeeded to break such injection of malicious codes by determining them in the early stages. Therefore the implementations of two machine learning methods are mandatory in the initial stages to achieve successful efficiency of the system. These two stages are primarily known as extracting relevant data from the raw dataset and selecting only the important ones so that it can be used for further classification.

The primary workflow of the system involves collecting and gathering the right amount of dataset available on various repositories. In my case, I have used the dataset from

Virus Share database. Once the dataset repository is finalized, the dataset is downloaded and further worked upon. Since this dataset contains large amount of irrelevant data and noisy data, hence its removal becomes mandatory in nature. Therefore, for this procedure to take place all relevant features are extracted from the repository and other irrelevant data is discarded. This is the most important step for any machine learning based model system. It is in this process that redundant data is removed so that it does not cause further efficiency reduction of the system. Determining appropriate feature extraction techniques become a major part in this stage of the implementation. In the next stage, all the extracted features are filtered and next only specific features are selected that would be fed into the machine learning algorithms. Therefore, in this stage as well selection of appropriate techniques for feature selection becomes a critical task. Once, the model undergoes the process of feature extraction and selection, the resulting dataset is then fed to train-test phase. In my case, I have utilized 80 percent of my dataset to train the algorithm and remaining 20 percent to test the model. After, the system model is trained and tested using certain pre- decided machine learning based algorithms; the model undergoes its evaluation so that its accuracy can be calculated. In my thesis, I have used the concepts of accuracy, precision, recall factors and ROC curves to determine the same.

The workflow of the proposed methodology involves the following stages as depicted in figure below:

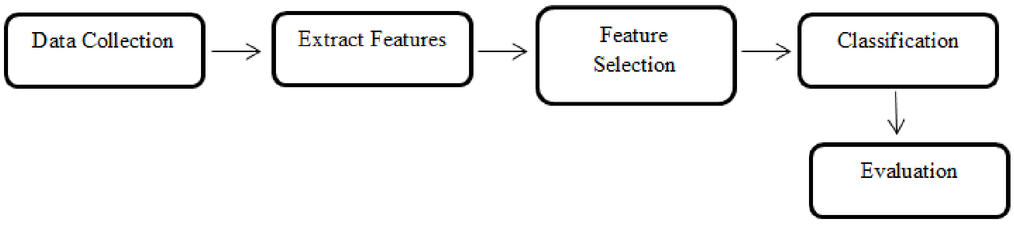


Figure 4: Workflow of the Proposed System

3.2 Dataset and Data Collection

The foremost step to implement my thesis is to collect the appropriate dataset required for accurate evaluation of machine learning algorithms so that the system could attain best optimized results. Therefore, the initial step of my thesis involved collecting raw data from Virus-share repository. This dataset contained a total of 7455 benign files and 12708 malicious files. In order to create and develop a clean dataset from the raw dataset, Windows 10 was installed on a virtual machine and PE files were collected from repositories such as .exe and

.dll. All the respective files were further copied in the respective directories of my system. Virus share was also used to obtain malware samples containing .exe and .dll PE files.

As mentioned earlier, the dataset used in my thesis is divided in the ratio of 80:20, with 80 percent used for training phase and rest 20 percent used for testing phase. Acquiring dataset and collecting relevant data is one of the most important key factors that results into achieving optimized results. Hence, it is necessary to search out for precise dataset that could help into accomplishing desired goals of developing a model.

The implementation of any machine learning based algorithm occurs by acquiring a data and later training an algorithm on this data. This is often referred to as the training dataset and thereby contains multiple features such as x entries and y labels. In such a scenario, a feature is said to be a variable that gives information about the entry. On the other hand, a label could be considered as something that needs to be predicted. Hence, when machine learning algorithms are used they tend to establish patterns and correlations between features so that accurate labels are predicted.

3.3 Data Pre-Processing

Once the raw data has been obtained from the repository, it is necessary to authenticate that the raw data is ready to be fed into the algorithm. However, the process of training is time consuming and forms a tedious task, but this step is necessary so that all the redundant and irrelevant featured from the dataset are removed. This process of filtering and selecting relevant features is often termed as feature selection from the dataset. However, different approaches can be used to extract and select best features from an algorithm using statistical tests to choose only the best k features.

3.4 Extracting Features

Once the data is collected, malicious PE files are later extracted for relevant features that could be significant for malware detection. This procedure is however carried out by using a built in tool of Python module known as pefile. Hence, in my thesis a total of 93 features were extracted. However, through literature survey it has been observed that many research scholars have used and implemented a variety of features to extract only the relevant features required to enhance the overall efficiency of the system.

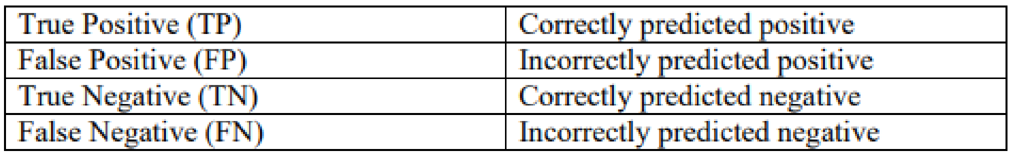
3.5 Feature Selection

Once the dataset is obtained from the repository, cleaned and filtered, it undergoes the process of extracting the most relevant features and thereby selecting them for further classification problem. Three of the most constant features from the PE header were removed and the final configuration size of the directory was set to 0 for all the samples from the dataset. Finally, all relevant features were selected and chosen to fed the machine for classification and prediction problem.

3.6 Evaluation Techniques

It is necessary to evaluate the systems model so that the overall efficiency of the model could be tracked upon. For this purpose, I have used precision. Recall factor, accuracy and ROC curves as a measure of evaluating them. The table below gives the possible predictions for evaluating the performance of the system:

Table3: Evaluation Parameters



True Positive (TP)	Correctly predicted positive
False Positive (FP)	Incorrectly predicted positive
True Negative (TN)	Correctly predicted negative
False Negative (FN)	Incorrectly predicted negative

Accuracy is one of the parameters used to determine the efficiency of the model. It is a method that tells the user the total number of correct and accurate predictions the resulting model has made. Whereas on the other hand, precision is referred to as the ability of the classifier to label a sample as either positive or negative. In addition to this, recall factors and its values are also considered to find all the positive values and outcomes in a resulting sample. Finally, the evaluating parameter of an ROC (Receiver Operating Characteristics) is used in a classification problem to measure the degree of separability and the probability of the curve.

4. CLASSIFICATION ALGORITHMS

4.1 AdaBoost

The abbreviation of AdaBoost stands for Adaptive Boosting and is implemented under machine learning algorithms (E.Raff,2017). It is considered to boost the normal working and implementation of an algorithm by majorly assigning weights. These weights are assigned to each and every formed observation which is required in the training phase. Once all the weights are assigned, all algorithmic models which are weak in nature are assigned boosted weights which help them to carry higher weights in comparison to their weights at origin. The implementation of this algorithm is further followed by performing iterations and gradually increasing the assigned weights in every iteration that it makes. Therefore, the concept of this algorithm helps to create decision based boundaries by making use of weak models to develop them into stronger ones. Hence, by using this method the overall accuracy of the system model is enhanced and the model is trained in such a way that it results into generating optimized precision with least number of errors so found (P.Vadrevu, 2016).

4.2 Decision Trees

The implementation of a decision tree also forms a part of supervised learning methodology wherein this algorithm can be used for classification as well as regression problems. The basic concept of this algorithm is built on a tree structured classifier that contains nodes within itself which are known to be as the features of the dataset (M.Polino, 2015). Further, the decision trees also have branches which are used to represent decision rules that would predict the final outcome of the model. However, in the visual representation of a decision tree, it primarily includes two nodes referred to as the decision node and leaf node. All the decisions with respect to multiple branches are taken by the decision node whereas the output of those decisions is given by the leaf nodes (N. Miramirkhani, 2017). This algorithm is majorly used in places where the model would either answer a “yes” or “no”. The diagram below briefs about the general structure of a decision tree:

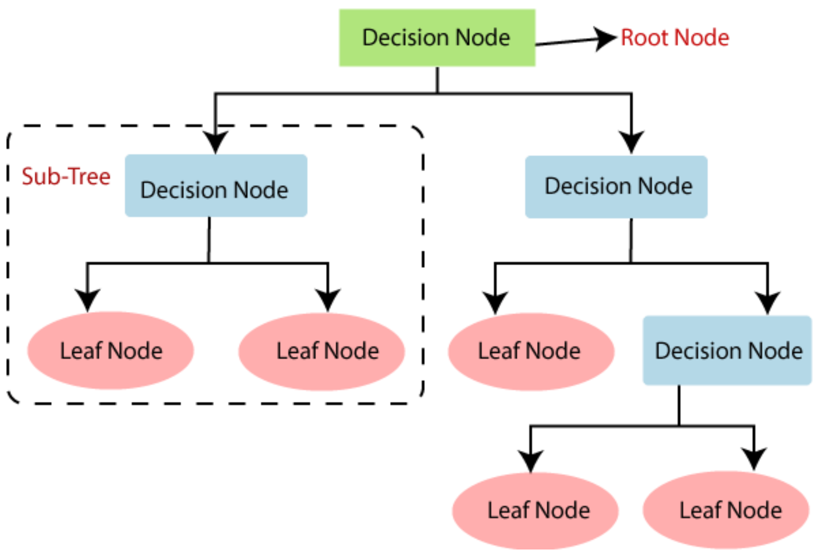


Figure 5: Visual representation of a decision tree (javatpoint.com)

4.3 Gradient Boosting

This algorithm is a type of machine learning algorithm that is commonly used for classification and regression to make predictions based on the model's system. This algorithm's working implementation consists of a set of weak estimating methods that are common in decision trees. Gradient Boosting combines a large number of weak classifiers to create a much stronger and larger dataset, which improves the prediction model's efficiency even more (T. Blazytko, 2017).

4.4 Random Forest

Random forest is considered to be as one of the most widely used machine learning algorithm that can implemented for classification and regression issues. However, its implementation is majorly based on the concepts of ensemble learning that involves the blend of multiple classifiers to solve a complex issue so that the overall efficiency of the system can be enhanced. It also comprises of various number of decision trees which are formed as subsets from original form of datasets so obtained from the repository. In the later stage, all the subsets are then combines together and an average of all the subsets are taken so as to predict the overall accuracy of the system (R. Jordaney, 2017). The diagram below illustrates the working of a random forest classifier.

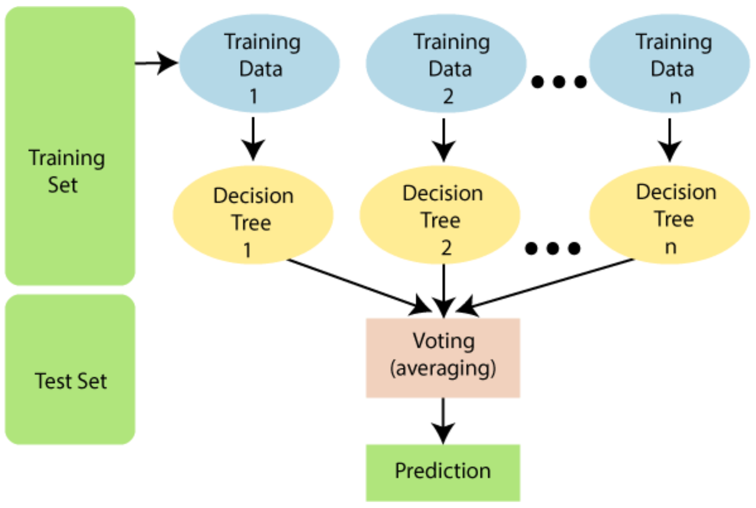


Figure 6: Visual representation of a random forest (javatpoint.com)

4.5 Logistic Regression

One of the most widely used machine learning algorithms is the implementation of logistic regression which comes under supervised learning technique. It is primarily used predicting final values of the output by making use of dependent variables from a given set of independent variables (K. Huang, 2009). Therefore, it is said that the final output of this machine learning algorithm is said to be categorical and discrete in nature. Hence, its final values are observed to be as either 0 or 1 or true and false. The implementation of a traditional logistic regression follows the basic concepts of a linear regression problem wherein a line is fitted between two axes and maximum values are derived from it. The major significance of a logistic regression is that, it possesses the capability to provide probabilities and later classifies those using discrete datasets (X. Hu, K. G. Shin, 2013).

4.6 CNN

In a traditional working of a neural network, it is rarely observed that the output from one phase is served as an input for another phase (P. O'Kane, 2013). Whereas, in real world applications majority of the outputs are dependent not only on external inputs but also on previous outputs. This concept of "persistence" is however not prevailed in neural networks. Hence, this inability of context based reasoning becomes a limitation of neural networks, and are therefore conceptualized using Convolutional Neural Networks. CNN's are a form of neural networks that alleviates the problem of NN's. Figure below depicts the working feedback loop of a typical CNN.

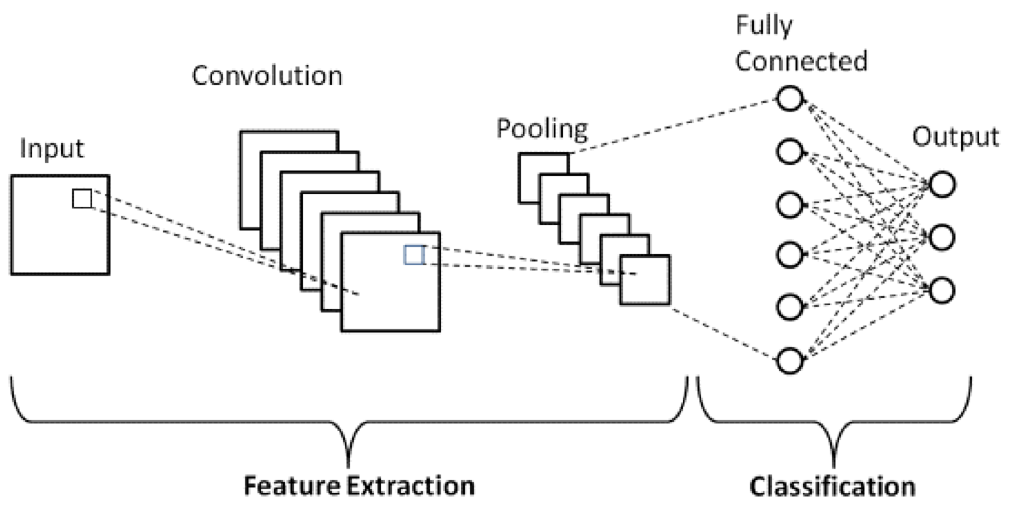


Figure 7: Visual representation of a CNN
(Van Hiep Phung, 2019)

4.7 LSTM

One of the most significant variant of an RNN network is the working model of an LSTM. An LSTM is primarily developed to handle the issues of long term dependencies. A conventional RNN is made of repeating modules that has a simple chain of module structure responsible to perform iterative actions. The repeating structure of an RNN is encapsulated as the *tanh* layer as depicted in figure below:

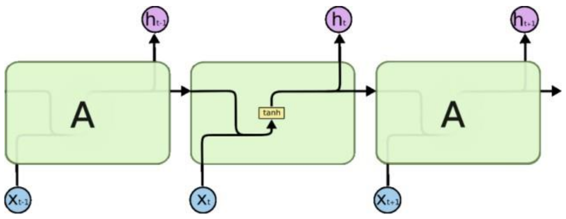


Figure 8: Repeating module of an RNN model (Achyut Ghosh, 2019)

LSTM's on the other hand, consists of four layers that are responsible for the iterative action in a neural network. All the layers of the network, represents a feature vector and acts like a linkage between input and output yields. Figure below shows the iterative structure of an LSTM:

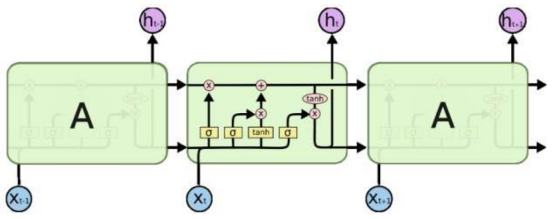


Figure 9: Repeating module of the LSTM model (Achyut Ghosh, 2019)

5. EXPERIMENTAL ANALYSIS AND RESULTS

This section of my thesis gives a brief description on the results obtained while conducting the experiments to gain optimized results. The chapter is divided into two sections with results being generated using machine learning and deep learning respectively. The thesis is however implemented using four machine learning algorithms namely:

- AdaBoost
- Gradient Boosting

- Random Forest
- Logistic Regression

Along with these four algorithms I have also proposed the implementation of stacking algorithm that makes use of Random Forest and AdaBoost as base classifiers and Decision Trees as the Meta classifier. The second half of the implementation is conducted using deep learning algorithms namely:

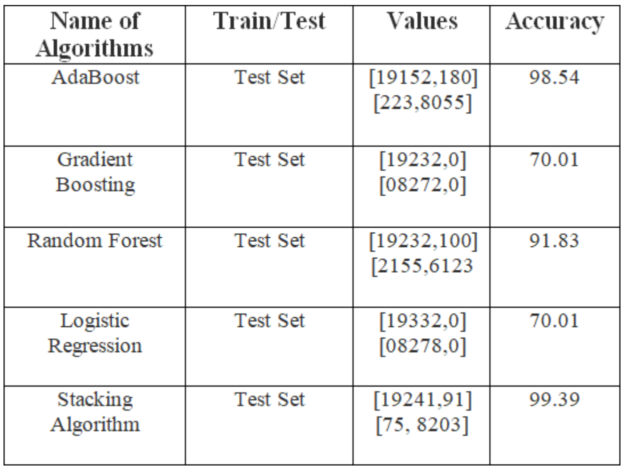
- CNN
- LSTM

The section below presents the output generated by individual algorithms and concludes an optimized method based on evaluation parameters such as accuracy, precision, recall factor, classification report and ROC curves. The final output of the experiment is also represented using GUI interface wherein the user comes to know the number of malicious and benign files so extracted.

5.1 Case Study1 – Using Machine Learning Algorithms

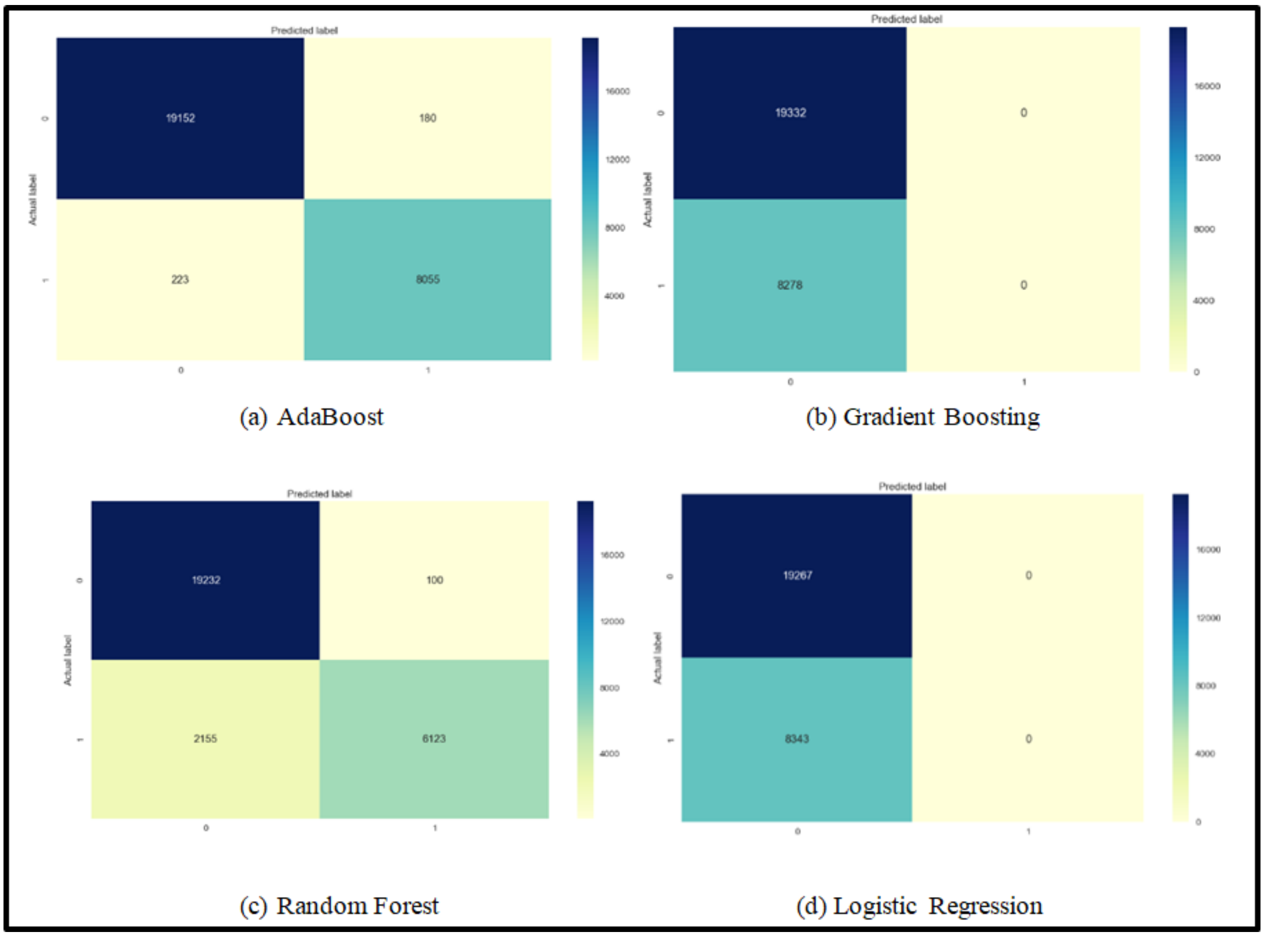
The primary aim of the study is to collect malicious files from a dataset repository and further classify the files as malicious or benign. Therefore, to successfully implement the purpose of the study, I have conducted experiments using four algorithms and one stacking based algorithm. The table below depicts the values obtained from AdaBoost, Gradient Boosting, Random Forest and Logistic Regression and Stacking algorithm:

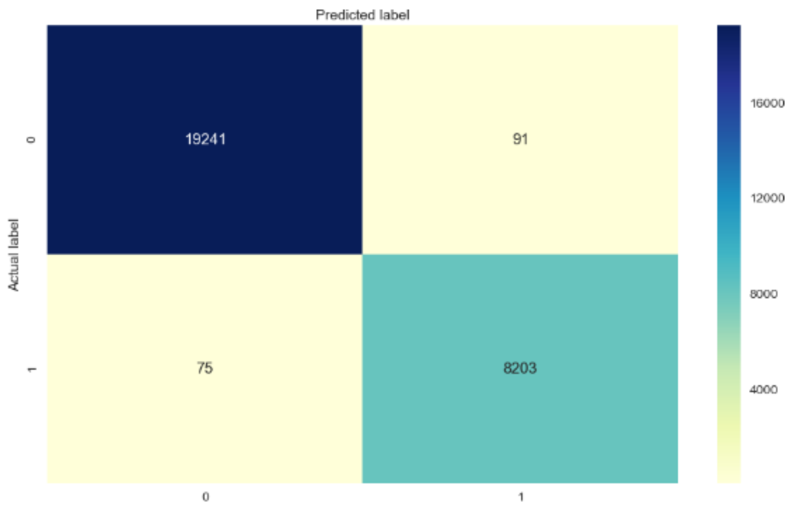
Table 4: Values obtained from confusion Matrix



Name of Algorithms	Train/Test	Values	Accuracy
AdaBoost	Test Set	[19152,180] [223,8055]	98.54
Gradient Boosting	Test Set	[19232,0] [08272,0]	70.01
Random Forest	Test Set	[19232,100] [2155,6123	91.83
Logistic Regression	Test Set	[19332,0] [08278,0]	70.01
Stacking Algorithm	Test Set	[19241,91] [75, 8203]	99.39

The figure below depicts the visual representation of a confusion matrix using four machine learning based algorithms and stacking algorithm:



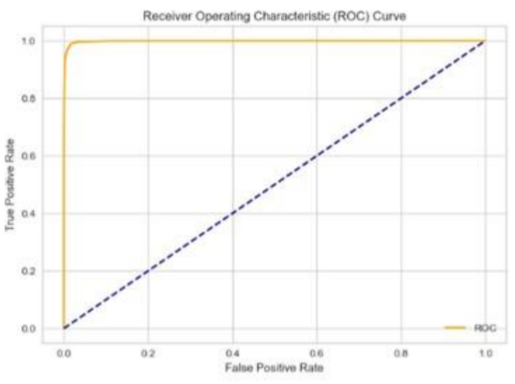


(e) Stacking Algorithm

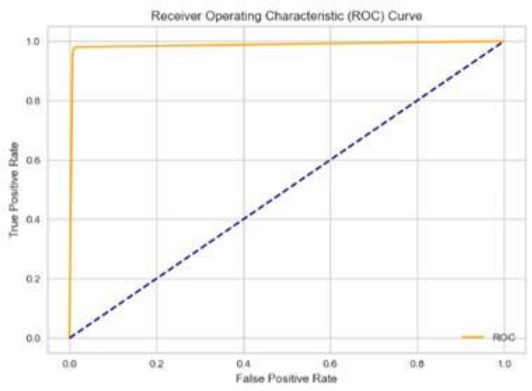
Figure 10: Visual Representation of a Confusion Matrix

The implementation of a confusion matrix helps to predict classification problems accurately. Its gives an illustration of predicted class to that of expected class values which are obtained during the testing phase of the model. The figure above depicts the confusion matrix so generated. It also gives information on false positive and true negative cases. In the above figure (a) provides the values obtained from AdaBoost method that generates 19152 cases of true positive and 180 cases of false positives for predicting malicious and benign files. On the other hand, the Gradient Boosting method generates 19332 false negatives and 0 true negatives cases malicious files. Whereas the implementation of Random Forest generates 19232 cases of true positive and 100 cases of false positives, followed by results obtained from logistic regression that generates 19267 cases of true positive and 0 cases of false positives. In a similar way all the values of various algorithms are generated using confusion matrix. However, it is important to note here that, the highest accuracy is being provided by the implementation of AdaBoost algorithm which generates an overall testing accuracy of 98.54 percent as observed from table 4. The final implementation of stacking algorithm generates 19241false negative cases and 91 true positives.

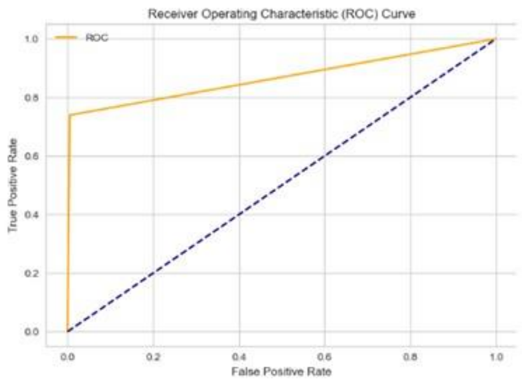
The figure below depicts the obtained ROC curves:



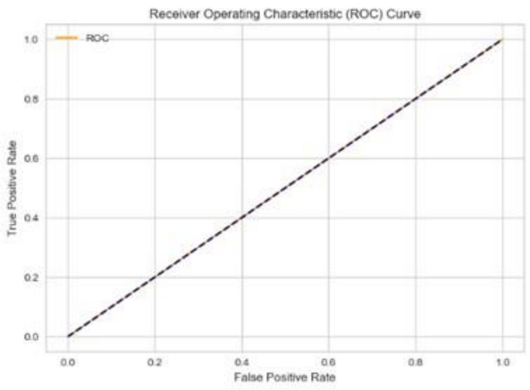
(a) AdaBoost



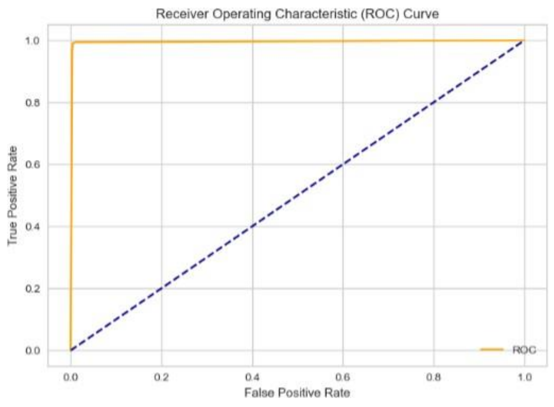
(b) Gradient Boosting



(c) Random Forest



(d) Logistic Regression



(e) Stacking Algorithm

Figure 11: Output of ROC curves

The values obtained from ROC curves are 0.99, 0.96, 0.87, 0.50 and 1.00 from AdaBoost, Gradient Boosting, Random Forest, Logistic Regression and Stacking algorithm respectively.

The table below depicts accuracy values obtained from classification reports:

Table5: Obtained values from Classification Report

Name of Algorithm	**Precision**		**Recall**		**FI-Score**	
AdaBoost	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.99	0.98	0.99	0.97	0.99	0.98
Accuracy	**0.98**					
Gradient Boosting	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.70	0.00	1.00	0.00	0.82	0.00
Accuracy	**0.70**					
Random Forest	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.90	0.98	0.99	0.74	0.94	0.84
Accuracy	**0.92**					
Logistic Regression	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.70	0.00	1.00	0.00	0.82	0.00
Accuracy	**0.70**					
Stacking Algorithm	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	1.00	0.99	1.00	0.99	1.00	0.99
Accuracy	**0.99**					

The overall results are thereby represented with their respective accuracy values. However, it has been observed that the highest accuracy has been generated when the model is experimented using stacking algorithm and thereby provides an accuracy of 99%.

5.2 Case Study2 – Using Deep Learning Algorithms

The primary aim of the study is to collect malicious files from a dataset repository and further classify the files as malicious or benign. Therefore, to successfully implement the purpose of the study, I have conducted experiments using two deep learning based algorithms. The figure below depicts the visual representation of a confusion matrix using two deep learning based algorithms:

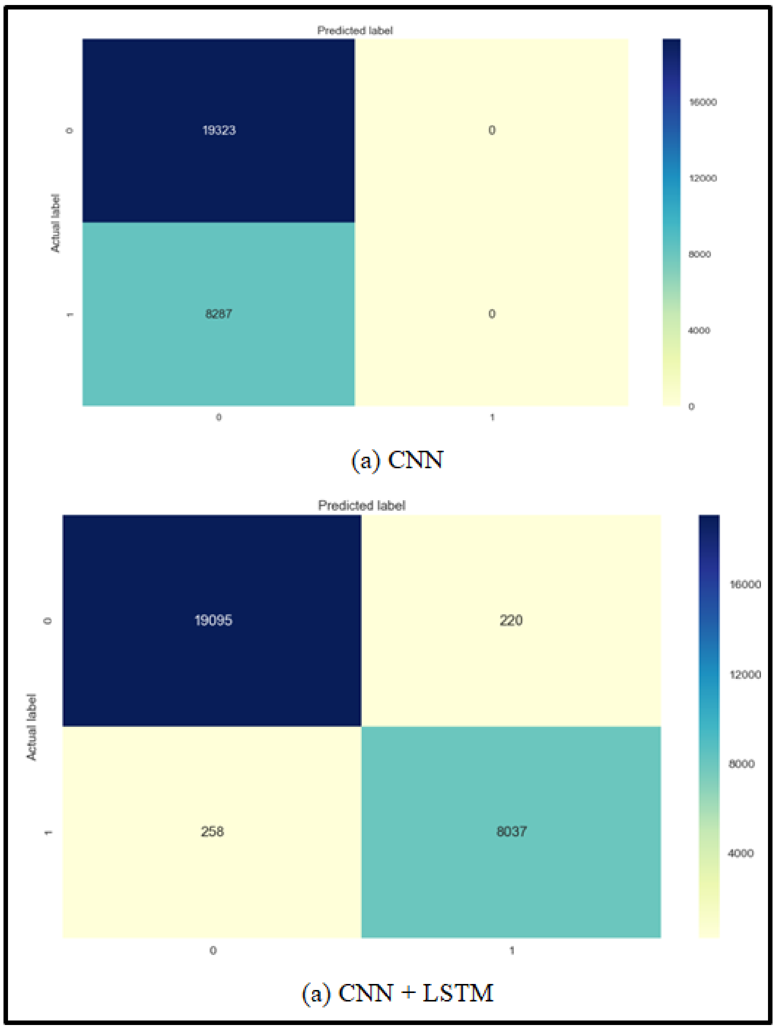


Figure 12: Visual Representation of a Confusion Matrix

The table below depicts the accuracy obtained from both the algorithms so used:

Table6: Values obtained from accuracy

Name of Algorithms	Train/Test	Accuracy
CNN	Test Set	69.98
CNN + LSTM	Test Set	98.26

The table below depicts accuracy values obtained from classification reports:

Table7: Obtained values from Classification Report

Name of Algorithm	**Precision**		**Recall**		**FI-Score**	
CNN	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.70	0.00	1.00	0.00	0.82	0.00
Accuracy	**0.70**					
CNN + LSTM	Positive Cases	Negative Cases	Positive Cases	Negative Cases	Positive Cases	Negative Cases
	0.99	0.97	0.99	0.97	0.99	0.97
Accuracy	**0.98**					

The overall results are thereby represented with their respective accuracy values. However, it has been observed that the highest accuracy has been generated when the model is experimented using CNN + LSTM algorithm and thereby provides an accuracy of 98%.

The figure below depicts the obtained ROC curves:

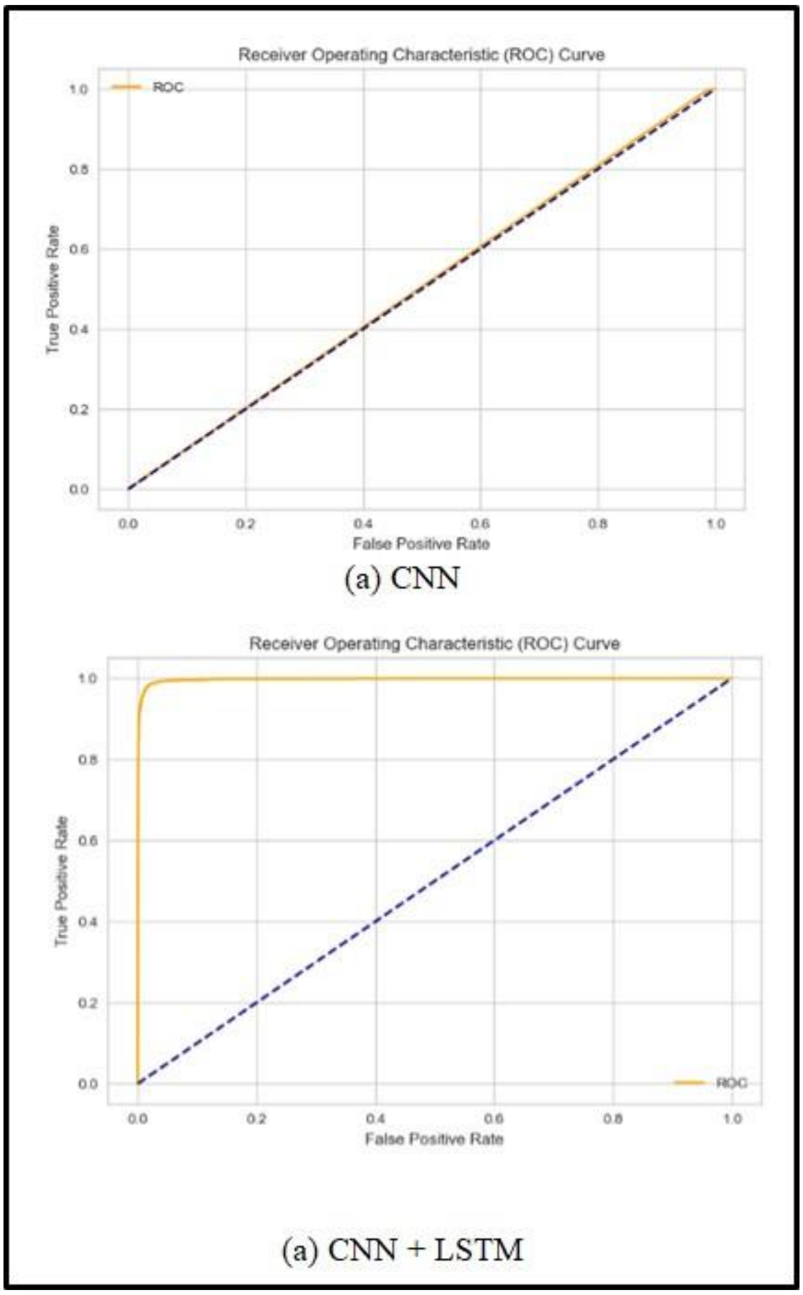
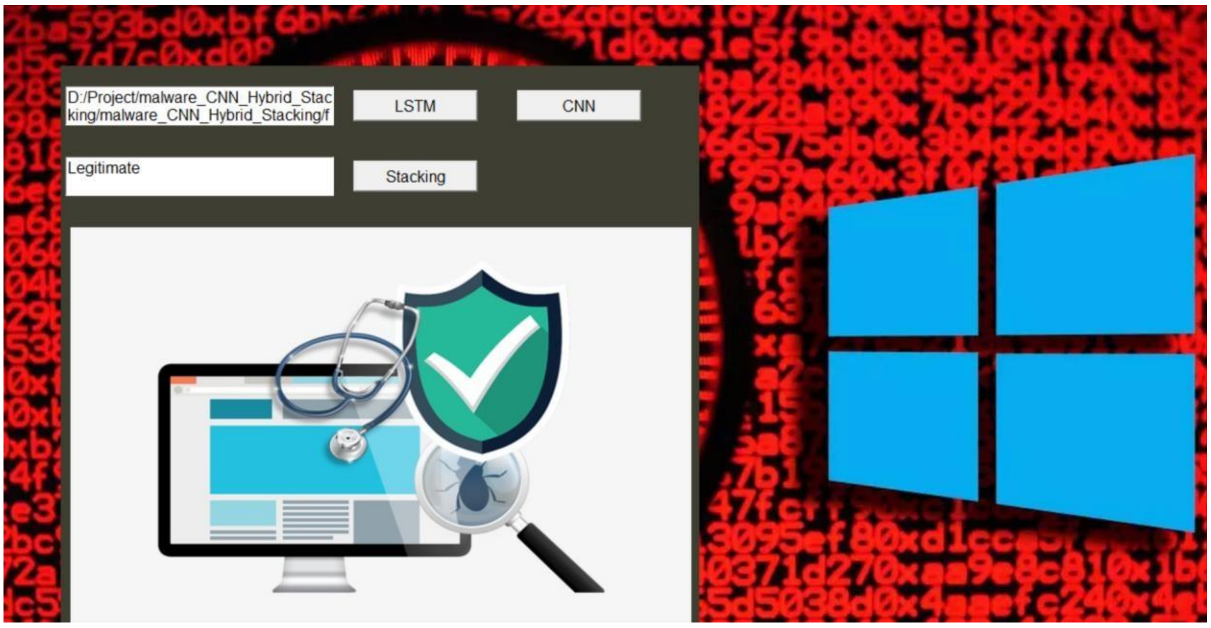


Figure 13: Output of ROC curves

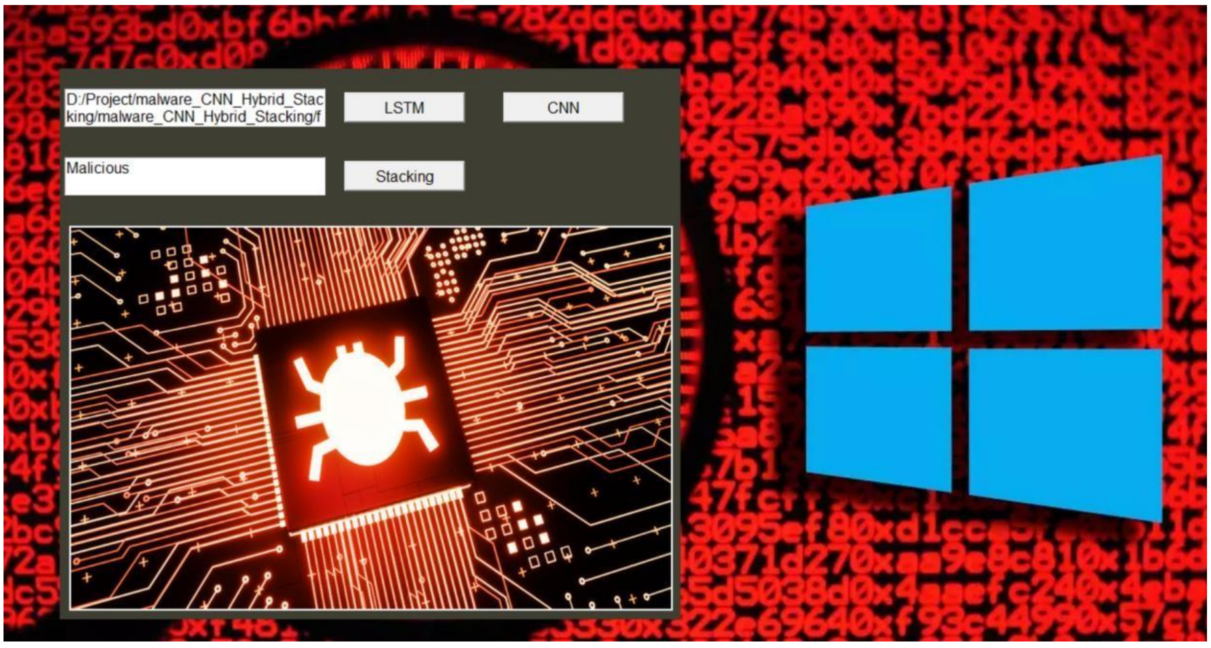
The values obtained from ROC curves are 0.51, 1.00 from CNN and CNN + LSTM respectively.

5.3 Case Study3 – Using GUI Interface

The deployment of the project model also occurs through a GUI interface wherein the user gets access to the application and can get information on the number of malicious and benign files on the basis of algorithm so used. The screenshots of GUI interfaces are depicted below:


D:/Project/malware_CNN_Hybrid_Stacking/malware_CNN_Hybrid_Stacking/f
LSTM
CNN
Legitimate
Stacking

Output of Screenshot 1 depicting malware files


D:/Project/malware_CNN_Hybrid_Stacking/malware_CNN_Hybrid_Stacking/f
LSTM
CNN
Malicious
Stacking

Output of Screenshot 2 depicting usage of deep learning algorithms

6. CONCLUSIONS

The fundamental aim of the thesis so proposed was to perform research on previous work being done by researchers and make use of machine learning and deep learning algorithms to detect malicious PE files and classify them as benign and good ware. For this purpose, relevant features were selected for attribute classification and the dataset was trained and tested on 80percent and 20 percent of the raw data so obtained from the repository. The implementation of the thesis was later followed by successfully gathering 20,000 PE files and extracting attributes from them so that they could further undergo the process of algorithmic implementation. In addition to his, I have implemented my thesis using four machine learning algorithms, one stacking algorithm and two deep learning models. amongst the four machine learning models, gradient boosting provided better accuracy, however, the implementation of stacking algorithm provided the highest accuracy of 99%. On the other hand, the experimental results so obtained after conducting deep learning models, it was observed that the combination of CNN with LSTM provided better results and accuracy in comparison to the implementation of CNN alone. Finally, in the thesis I have provided a GIU interface that lets the user interact and obtain results with respect to the malicious files.

LIMITATIONS:

However, the presented research also encounters certain limitations and drawback. The major drawback was to get access to malware dataset so that analysis could further be performed. Since, data collection possessed to become a challenge, the later stages which involved filtering samples and variants became difficult. Another major drawback was with the samples being provided on online repositories. All the samples were a mix of malware and good ware, with some of them being inconsistent and not being labelled. In addition to this, another limitation that added to the thesis was the sample dataset being quite old to perform upon and not being of recent values. Hence, this proved to be a hurdle in implementing the same.

FUTURE WORK

In the future, the same working model can be extended by adding multiple features to optimize complex models with highest possible accuracy. In addition to this, the same model can be tested and implemented using various other deep learning and machine learning algorithms that could in turn benefit the model by making it fully automated in nature while using less features and attaining high accuracy.

REFERENCES

1. International Telecommunication Union. Statistics. Available online: https://www.itu.int/en/ITU-D/Statistics/Pages/ publications/yb2018.aspx (accessed on 20 November 2019)
2. Namanya, A.P., Cullen, A., Awan, I.U. and Disso, J.P., 2018, August. The world of Malware: An overview. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* (pp. 420-427). IEEE.
3. Azeez, N.A., Salaudeen, B.B., Misra, S., Damaševičius, R. and Maskeliūnas, R., 2020. Identifying phishing attacks in communication networks using URL consistency features. *International Journal of Electronic Security and Digital Forensics*, *12*(2), pp.200-213.
4. Yong, B., Wei, W., Li, K.C., Shen, J., Zhou, Q., Wozniak, M., Połap, D. and Damaševičius, R., 2020. Ensemble machine learning approaches for webshell detection in Internet of things environments. *Transactions on Emerging Telecommunications Technologies*, p.e4085.
5. Boukhtouta, A., Mokhov, S.A., Lakhdari, N.E., Debbabi, M. and Paquet, J., 2016. Network malware classification comparison using DPI and flow packet headers. *Journal of Computer Virology and Hacking Techniques*, *12*(2), pp.69-100.
6. Odusami, M., Abayomi-Alli, O., Misra, S., Shobayo, O., Damasevicius, R. and Maskeliunas, R., 2018, November. Android malware detection: A survey. In *International conference on applied informatics* (pp. 255-266). Springer, Cham.
7. Bazrafshan, Z., Hashemi, H., Fard, S.M.H. and Hamzeh, A., 2013, May. A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology* (pp. 113-120). IEEE.
8. Souri, A. and Hosseini, R., 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-centric Computing and Information Sciences*, *8*(1), pp.1-22.
9. Ye, Y., Li, T., Adjeroh, D. and Iyengar, S.S., 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, *50*(3), pp.1-40.
10. Azeez, N.A., Odufuwa, O.E., Misra, S., Oluranti, J. and Damaševičius, R., 2021, March. Windows PE malware detection using ensemble learning. In *Informatics* (Vol. 8, No. 1, p. 10). Multidisciplinary Digital Publishing Institute.
11. M.G. Schultz, E. Eskin, F. Zadok, S.J. Stolfo, Data mining methods for detection of new mali-cious executables, in Proceedings 2001 IEEE Symposium on Security and Privacy. S&P(2001).IEEE (2001), pp. 38–49
12. Anderson, B., Quist, D., Neil, J., Storlie, C. and Lane, T., 2011. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*, *7*(4), pp.247-258.

13. Eskandari, M., Khorshidpour, Z. and Hashemi, S., 2013. HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection. *Journal of Computer Virology and Hacking Techniques*, *9*(2), pp.77-93.
14. Khodamoradi, P., Fazlali, M., Mardukhi, F. and Nosrati, M., 2015, October. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. In *2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS)* (pp. 1-6). IEEE.
15. LeDoux, C. and Lakhotia, A., 2015. Malware and machine learning. In *Intelligent Methods for Cyber Warfare* (pp. 1-42). Springer, Cham.
16. Liang, G., Pang, J. and Dai, C., 2016. A behavior-based malware variant classification technique. *International Journal of Information and Education Technology*, *6*(4), p.291.
17. Ucci, D., Aniello, L. and Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, pp.123-147.
18. Gandotra, E., Bansal, D. and Sofat, S., 2016, December. Zero-day malware detection. In *2016 sixth international symposium on embedded computing and system design (ISED)* (pp. 171-175). IEEE.
19. Damodaran, A., Troia, F.D., Visaggio, C.A., Austin, T.H. and Stamp, M., 2017. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(1), pp.1-12.
20. Mirza, Q.K.A., Awan, I. and Younas, M., 2018. CloudIntell: An intelligent malware detection system. *Future Generation Computer Systems*, *86*, pp.1042-1053.
21. Wang, W., Zhao, M. and Wang, J., 2019. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing*, *10*(8), pp.3035-3043.
22. Zhang, H., Zhang, W., Lv, Z., Sangaiah, A.K., Huang, T. and Chilamkurti, N., 2020. MALDC: a depth detection method for malware based on behavior chains. *World Wide Web*, *23*(2), pp.991-1010.
23. Mirza, Q.K.A., Awan, I. and Younas, M., 2018. CloudIntell: An intelligent malware detection system. *Future Generation Computer Systems*, *86*, pp.1042-1053.
24. Ma, Z., Ge, H., Liu, Y., Zhao, M. and Ma, J., 2019. A combination method for android malware detection based on control flow graphs and machine learning algorithms. *IEEE access*, *7*, pp.21235-21245.
25. Ren, Z., Wu, H., Ning, Q., Hussain, I. and Chen, B., 2020. End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Networks*, *101*, p.102098.
26. Yuxin, D. and Siyi, Z., 2019. Malware detection based on deep learning algorithm. *Neural Computing and Applications*, *31*(2), pp.461-472.
27. Pei, X., Yu, L. and Tian, S., 2020. AMalNet: A deep learning framework based on graph convolutional networks for malware detection. *Computers & Security*, *93*, p.101792.
28. Anderson, B., Quist, D., Neil, J., Storlie, C. and Lane, T., 2011. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*, *7*(4), pp.247-258.
29. Raff, E. and Nicholas, C., 2017, August. An alternative to NCD for large sequences, Lempel-Ziv Jaccard distance. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1007-1015).
30. Vadrevu, P. and Perdisci, R., 2016, May. Maxs: Scaling malware execution with

sequential multi-hypothesis testing. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (pp. 771-782).

31. Polino, M., Scorti, A., Maggi, F. and Zanero, S., 2015, July. Jackdaw: Towards automatic reverse engineering of large datasets of binaries. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 121-143). Springer, Cham.
32. Miramirkhani, N., Appini, M.P., Nikiforakis, N. and Polychronakis, M., 2017, May. Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts. In *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 1009-1024). IEEE.
33. Blazytko, T., Contag, M., Aschermann, C. and Holz, T., 2017. Syntia: Synthesizing the semantics of obfuscated code. In *26th USENIX Security Symposium (USENIX Security 17)* (pp. 643-659).
34. Jordaney, R., Sharad, K., Dash, S.K., Wang, Z., Papini, D., Nouretdinov, I. and Cavallaro, L., 2017. Transcend: Detecting concept drift in malware classification models. In *26th USENIX Security Symposium (USENIX Security 17)* (pp. 625-642).
35. Huang, K., Ye, Y. and Jiang, Q., 2009, August. Ismcs: an intelligent instruction sequence-based malware categorization system. In *2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication* (pp. 509-512). IEEE.
36. Hu, X., Shin, K.G., Bhatkar, S. and Griffin, K., 2013. {MutantX-S}: Scalable Malware Clustering Based on Static Features. In *2013 USENIX Annual Technical Conference (USENIX ATC 13)* (pp. 187-198).
37. O'Kane, P., Sezer, S., McLaughlin, K. and Im, E.G., 2013. SVM training phase reduction using dataset feature filtering for malware detection. *IEEE transactions on information forensics and security*, *8*(3), pp.500-509.
38. emmanuel masabo et al. a state-of-the-art survey on polymorphic malware analysis and detection techniques.
39. https://www.vox.com/new-money/2017/6/29/15888570/petya-ransomware-attack-explained
40. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
41. https://www.javatpoint.com/machine-learning-random-forest-algorithm
42. Van Hiep Phung, Eun Joo Rhee, A high accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets, mdpi, 2019
43. Stock Price Prediction Using LSTM on Indian Share Market A. Ghosh et al