National College of Ireland

Detecting Malware Based on Portable Executable Analysis

MSc Research Project
M.Sc. Cybersecurity

Shubham Pandharpote
Student ID: x20143877

School of Computing
National College of Ireland

Supervisor: Liam McCabe

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shubham Prashant Pandharpote
Student ID:	x20143877
Program:	M.Sc. Cybersecurity **Year:** 2021-2022
Module:	MSc Internship
Supervisor:	Liam McCabe
Submission Due Date:	16/12/2021
Project Title:	Detecting Malware Based on Portable Executable Analysis
Word Count:	16 **Page Count: 5304**

I hereby certify that the information contained in this (my submission) is research information I conducted for this project. All information other than my contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other authors' written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shubham Prashant Pandharpote

Date: 16/12/2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on the computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Acronyms

Original Form	Acronym
DLL	Data Linked Libraries
PCA	Principal Component Analysis
SVM	Support Vector Machine

Table of Figures

Table of Tables

Detecting Malware Based on Portable Executable Analysis

Shubham P. Pandharpote
X20143877

ABSTRACT

The antivirus software work on the principle of detecting the virus based on the signatures. However, the malware developers have developed more powerful these days for which sometimes malware detection based on signature becomes difficult. To tackle this problem, the system designed extracts the features from Portable executables, which are analyzed with the help of machine learning techniques. The portable executable is a file format for executables, object code, Data Link Library (DLLs), and other portable executives used in 32 bits and 64 bits versions of Windows Operating systems. The paper is regarding the detection of Malware by analyzing portable executable files with the help of Machine Learning Techniques. The dataset is used which consists of malicious portable executable files. Machine Learning Techniques, Support Vector Machine (SVM), and Gradient Boosting are being used to train the structure by extracting features in such a way that a particular file is being detected after feature extraction. The features extracted from the dataset are Optional Header and Section Header. After the implementation process was carried out, based on it, accuracy was calculated.

Keywords: Malware, Portable Executables, Windows Operating System, Support Vector Machine (SVM), Gradient Boosting.

1 Introduction:

Malware is also referred to as malicious software. Malware is designed with the thought of designing stealing information, financial gains such as encrypting files and demanding some amount in normal currency format or cryptocurrency format for decrypting the files of a victim. Furthermore, malware is designed to infect systems such as servers of IT giant companies for stealing confidential information. Over the period malware has become more powerful. In addition to that, malware developers or hackers are either polymorphing the preceding versions of malware or adding some extra functionalities which are making malware more powerful. Malware is classified into various types based on its function. Malware is analyzed as static or dynamic analysis. (What is Malware Analysis? | Different Types of Malware Analysis, 2021)

In static analysis malware code is analyzed to get knowledge about the functioning of malware. Accordingly, developers of virus detecting software or IT teams will add security aspects to their environment. This also impacts the performance of a dynamic analysis. (What is Malware Analysis? | Different Types of Malware Analysis, 2021)

In dynamic analysis application or entry point of how malware is carried on is analyzed. In addition, what changes are made to underlying systems are made. Mostly altering the

underlying system is done by the malware to enter the system as and when required. In the normal system, if changes are done to the underlying system warnings occur describing what changes are done to it. In a malware-free system when various activities are carried out such as new services are installed, other behavioral-related changes are notified, changes occurring to network traffic are analyzed. (What is Malware Analysis? | Different Types of Malware Analysis, 2021)

Malware has become more focused that their signatures are not detected by a virus detecting software. This malware is old programs with new signatures, but these are designed in such a way that they are either polymorphic or metamorphic. Such viruses are undetected with their current signatures. Malware detection is done currently based on the signatures of the malware. For example, if there is an abnormal behavior found in a system, then users will use virus detecting software to scan the system. This virus detecting software will search for a known signature with the ones which are stored in the database. After this stage, if the virus detecting software can find any familiar signature in the system, then it will either delete or quarantine it. (What Is Signature-Based Malware Detection? 2021)

There is another method of detecting malware, wherein the malware is detected by analyzing the features. This method is known as the heuristic method. In this method, if anti-virus software finds any malware it will work on the features of malware thereby decompiling it. While testing the software, it is either tested on the real-world scenario or in a virtualized environment by decompiling it by following certain steps the malicious software is eradicated from the system. In addition to it, as it thoroughly analyses the malware, it also monitors its various ill – behaviors like self–replication, altering the system, overwriting the files, and various other actions are monitored. (Cyber Security Resource Center for Threats & Tips | Kaspersky, 2021)

Windows is easy to use an operating system that is used worldwide, right from creating reports in small offices up to the computation in large companies for designing and testing the software. Windows has created its data structure and its files are known as portable executable files. As mentioned in the introduction of malware, attackers have become very focused and they are developing the malware for a specific purpose, one such purpose is compromising windows systems using portable executables. Consider a scenario, wherein an attacker will mold an important portable executable file with malware and make it a carrier for its malware. When an application is installed in any system which consists of this file, while execution call will reach up to that portable executable file, with the execution of that portable executable the malware will also be executed, and it will help the attacker in the creation of backdoor in the system. This backdoor will be helpful for the attacker to perform malicious activities. Thus, it is necessary to analyze the portable executable. Portable executables have various features from which in this research two features will be analyzed based on the dataset. Machine learning techniques will be used for analyzing the features of portable executables.

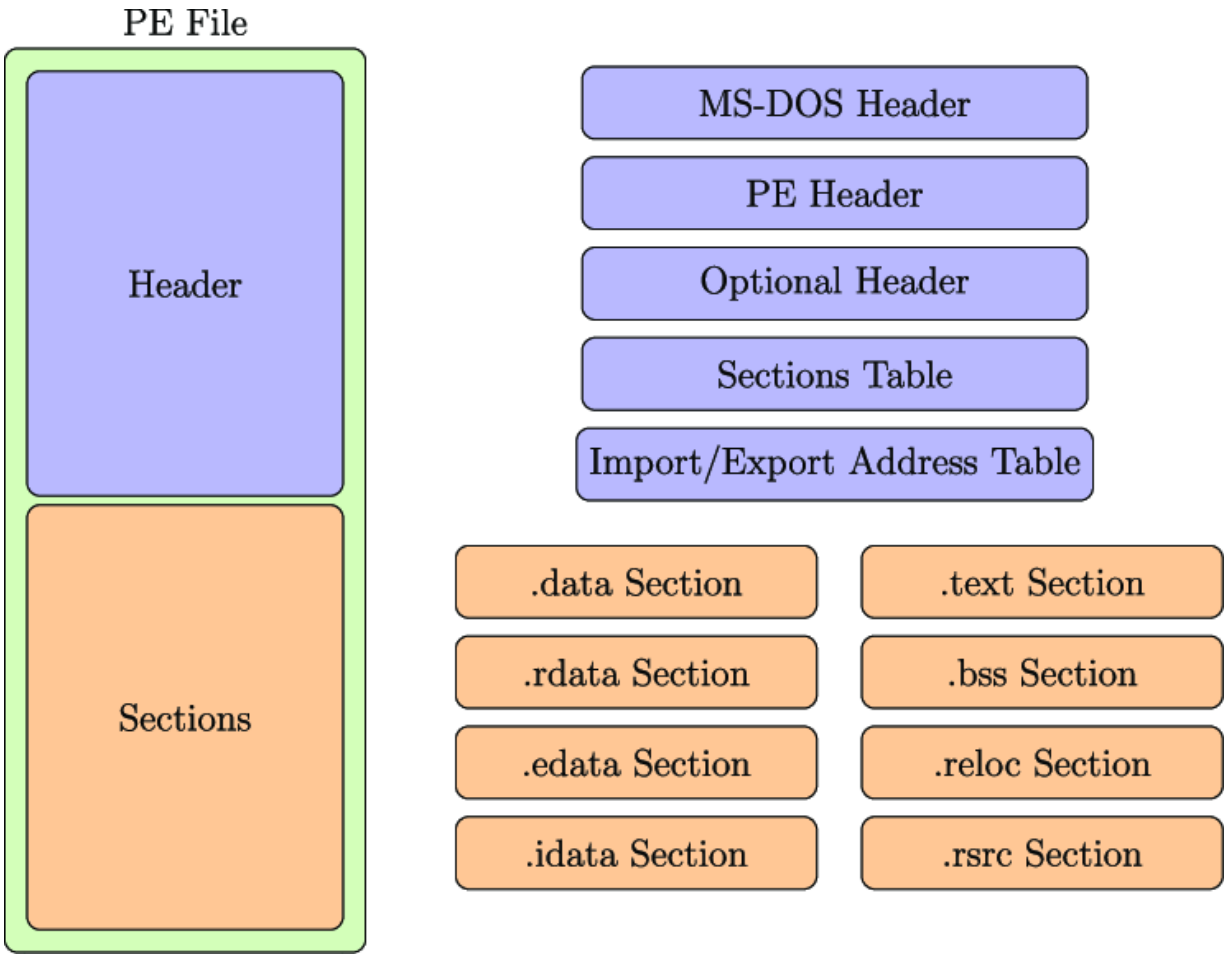

Fig.1: Architecture of Portable Executable File.

2 Research Question:

RQ: Is it possible for machine learning algorithms to accurately detect the malware?
Description: The dataset which will be consisting of portable executables will be analyzed by using machine learning algorithms. The results obtained from this analysis will help detect the malware. According to research, it is found that various machine learning algorithms are used to detect the malware from datasets and the results found from it were not as expected. So, by using the algorithms Support Vector Machine(SVM) and Gradient Boosting the proper results will be found. The outcomes from these results obtained will help detect the malware.

3. Literature Review:

In the paper, PE header analysis for Malware Detection, the author Samuel Kim, used machine learning algorithms in the research and identified malware by analyzing Portable Executables. In algorithm Support Vector Machine (SVM), wherein, classification is done based on the best line or hyperplane between two classes. There is a generation of hyperplane which by determining the widest margin which further divides two groups without having any point inside the margin. During the research, the accuracy achieved for the free approach was 80% and the accuracy achieved for a feature-full was 57%. Further, in an experiment the accuracy achieved was 96% for a feature-full, and feature-free was 99%. Further in the research author has mentioned Portable Executables (PE), how were the malware samples achieved, and from where the dataset was achieved. In addition to previous information, the

number of malware samples was discussed in the paper by the author. The information mentioned is useful for our research. (PE header analysis for Malware Detection, 2018)

In the paper, Malware Detection System Based on an In-depth Analysis of the Portable Executable Headers, authors Mohamed Belaoued, Bouchra Guelib, Yasmine Bounaas, Abdelouahid Derhab, and Mahmoud Boufaida made use of Portable executable header fields during the research. Some Portable Executable features were extracted from malware samples. Some features amongst that were Optional Header and Section Headers were extracted. Authors made use of machine learning algorithms, out of which algorithm was a Support Vector Machine (SVM). The dataset was used for malware detection from malicious portable executables. The accuracy found was between 99% to 100%. From this paper the information regarding machine learning algorithms and information regarding Optional Header and Section Header is useful. (Malware_Detection_System_Based_on_an_Indepth_Analysis_of_the_Portable_Executable_Headers, 2021)

In the paper, Learning the PE Header, Malware Detection with Minimal Domain Knowledge, authors Edward Raff, Jared Sylvester, Charles Nicholas made use of Portable Executable Headers and extracted certain features were extracted. Initial extraction was done using Python Library, by the authors. Some features were extracted from the malware sample of Portable Executables, one of which was the Optional Header. In the paper, the author has mentioned that feature full and feature free approaches gave accuracy for 85% and 97% respectively. The information regarding the Optional header and Portable Executables(PE) is useful from this paper. (Learning the PE Header, Malware Detection with Minimal Domain Knowledge, 2021)

In the paper, Windows Portable Executor Malware detection using Deep learning approaches, author Yogesh Bharat Parmar has made use of portable executables, features of portable executables are extracted using machine learning algorithms. Algorithms are used to detect the malicious behavior of Portable Executables. Initially, the author has shown the model for the proposed system. Further, in the paper, the author has discussed from what source the data has been collected and details regarding the dataset. Further how data pre-processing is done and what is its importance in the overall process is explained. In the feature engineering process, the author has mentioned two important processes feature extraction and feature selection. In model training, the author has explained the use of machine learning algorithms which the author has used during research. Accuracy research during research by the author was 94%. (Windows Portable Executor Malware detection using Deep learning approaches - NORMA@NCI Library, 2021)

Author Mayuri Wadkar, in her research paper Detecting Malware Evolution Using Support Vector, used only Support Vector Machine during the research for identifying malware using dataset. The features extracted from the dataset were section header, which is one of the aspects of Portable Executables. Various Portable Executable (PE) features were further extracted during the feature extraction. Furthermore, the author mentioned about what was

the purpose of utilizing the Support Vector Machine (SVM) algorithm in the research. Moreover, the author also mentioned how it was best suited in the research. In a further, go the author used chi-square analysis for Portable Executables features which were extracted. In this paper author didn't find any accuracy however in future work it is explained that a Support Vector Machine can be used to detect malware and find the accuracy for detecting the malware. This information regarding the usage of Portable Executables (PE) and Support Vector Machine (SVM) is useful. (Detecting malware evolution using support vector machines, 2019)

Authors, Tzu-Yen Wang, Chin-Hsiung Wu, Chu-Cheng Hsieh, researched the topic of Detecting Unknown Malicious Executables Using Portable Executable Headers and identified malicious executables using certain machine learning algorithms. In this research, the authors have discussed Portable Executables how it is analyzed for malware detection. In further part, authors have discussed the Malware Detection Model built during the research for identifying unknown malicious portable executables, wherein authors have discussed steps for overall research. In data collection, the authors have discussed is how was data collected and from where it was collected. On the further go, the authors have mentioned the breakdown of the data. The further step taken was data extraction. In this, the authors performed data extraction with the help of certain tools and further steps performed. In addition to it, a table of extracted features from the data set was displayed. Further steps were carried out for data extraction using the Support Vector Machine (SVM) algorithm. In detail research regarding how the algorithm was used and how it works was explained. After the experiment is performed, the overall accuracy that is achieved was approximately 94%. The information about Portable Executables and Support Vector Machine algorithm used is useful in research. (New Approach for Detecting Unknown Malicious Executables, 2010)

Authors, Huu-Danh Pham, Tuan Dinh Le, and Thanh Nguyen Vu in their paper Static PE Malware Detection Using Gradient Boosting Decision Trees Algorithm have identified the anomalies in the dataset using Gradient Boosting Algorithm. The dataset used by authors consisted of Portable Executables which were used for detecting the malware. The algorithm Gradient Boosting was used by researchers as it consumes less time for training the model and gives out impressive results for detecting anomalies. Further in the paper researchers have discussed feature engineering, wherein researchers have discussed that they had to remove the features which had missing and noisy data. As, these missing and noisy features could have caused distortion, resulting in less accuracy of detection of malware. The information mentioned was useful while feature engineering in the project and also for the creation of models. (Static PE Malware Detection Using Gradient Boosting Decision Trees Algorithm, 2021)

3 Research Methodology

The research proposed will have various stages of the methodology. The stages will consist of the collection of a topic-relevant dataset, next is dataset cleaning by removing blank fields and irrelevant fields from the data. After cleansing of data features will be extracted, then in

later stage required fields will be selected. In a later stage, data processing will be done. Further stages will be creating a model using machine learning algorithms, after models are created dataset will be passed through models. When accuracies will be achieved from each of the models, the accuracies will be compared and a decision will be made, which algorithm gave the highest accuracy for detecting malware.

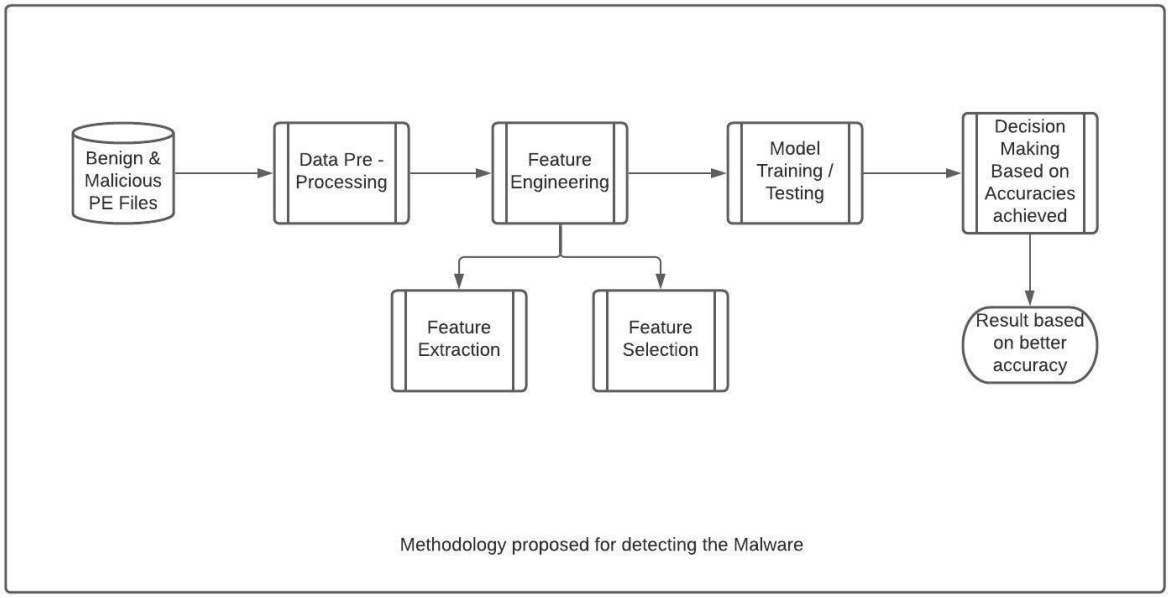

Fig.2: Methodology Proposed for Detecting the Malware.

3.1 Data Collection:

The dataset required for detecting malware will be taken from the website Kaggle. The dataset consisted of more than 10000 Windows Portable Executables' samples, wherein the sample can be either malware or benign. Dataset consisted of 79 features however, only a few features will be selected for analysis. AddressOfEntryPoint, BaseOfCode, ImageBase, SectionAlignment, DllCharacteristics, 'SectionsLength', SectionEntropy, SectionRawSize, SectionVirtualSize are few of the features involved in the dataset.

3.2 Data Pre – Processing:

These are some of the important steps while dealing with a dataset. While going through a dataset, there is an occurrence of some missing cells or cells with improper values, due to which can result in less accuracy while detection of malware. Thus, it should be taken care of, by converting the values into relevant data – type or dropping that cell which can lead to increased accuracy. Furthermore, cells with no values in the dataset are dropped as well for increasing the accuracy.

3.3 Feature Engineering:

This stage consists of two substages wherein, feature extraction is the stage where all the features will be extracted from the dataset. Feature selection is a process where

features relevant for detecting the malware will be selected. Data consist of 79 features. Principal Component Analysis (PCA) will be used to change the dimensions of the dataset. By using Principal Component Analysis (PCA) the whole dataset will be converted into two columns, this conversion will reduce the time required for training the dataset and for testing of a dataset. Time required to perform an experiment and get results will also be reduced.

3.4 Model Training:

Models will be trained and tested for detecting the malware. During the experiment, two models Support Vector Machine (SVM) and Gradient Boosting for detecting malware. Dataset will be split into 80% and 20% whereas, 80% will be used while training and 20% will be used for testing purposes. Description for each model is elaborated below in the Design Specification block.

3.5 Evaluation of Results:

At this stage, various results will be analyzed with the help of performance metrics, which will include stages such as Precision, accuracy, recall, F1 – score. These matrices will be calculated for every model. All the matrices are defined below:

1. Precision:

Precision is defined as,

Precision

= True Positives / True Positives + False Positives

= True Positives / Predicted True

Furthermore, when the model is positive, it will collect its accuracy of it. The models will higher precision are more likely to produce true results, than those false results. (LogisticRegressionPart2, 2021)

2. Recall:

The recall is defined as,

Recall

= True Positives / True Positives + False Negatives

= True Positives / Actually True

Based on all true samples recall collects all the abilities of a model predicting true. (LogisticRegressionPart2, 2021)

3. F1 – Score:

F1 – Score is the harmonic mean of precision and recall. Combining false positive and false negative results will calculate F1 – Scores. Further, it can be defined as,

F1 – Score = 2 * ((Precision * Recall) / (Precision + Recall)) (Long Short-Term Memory, 2021)

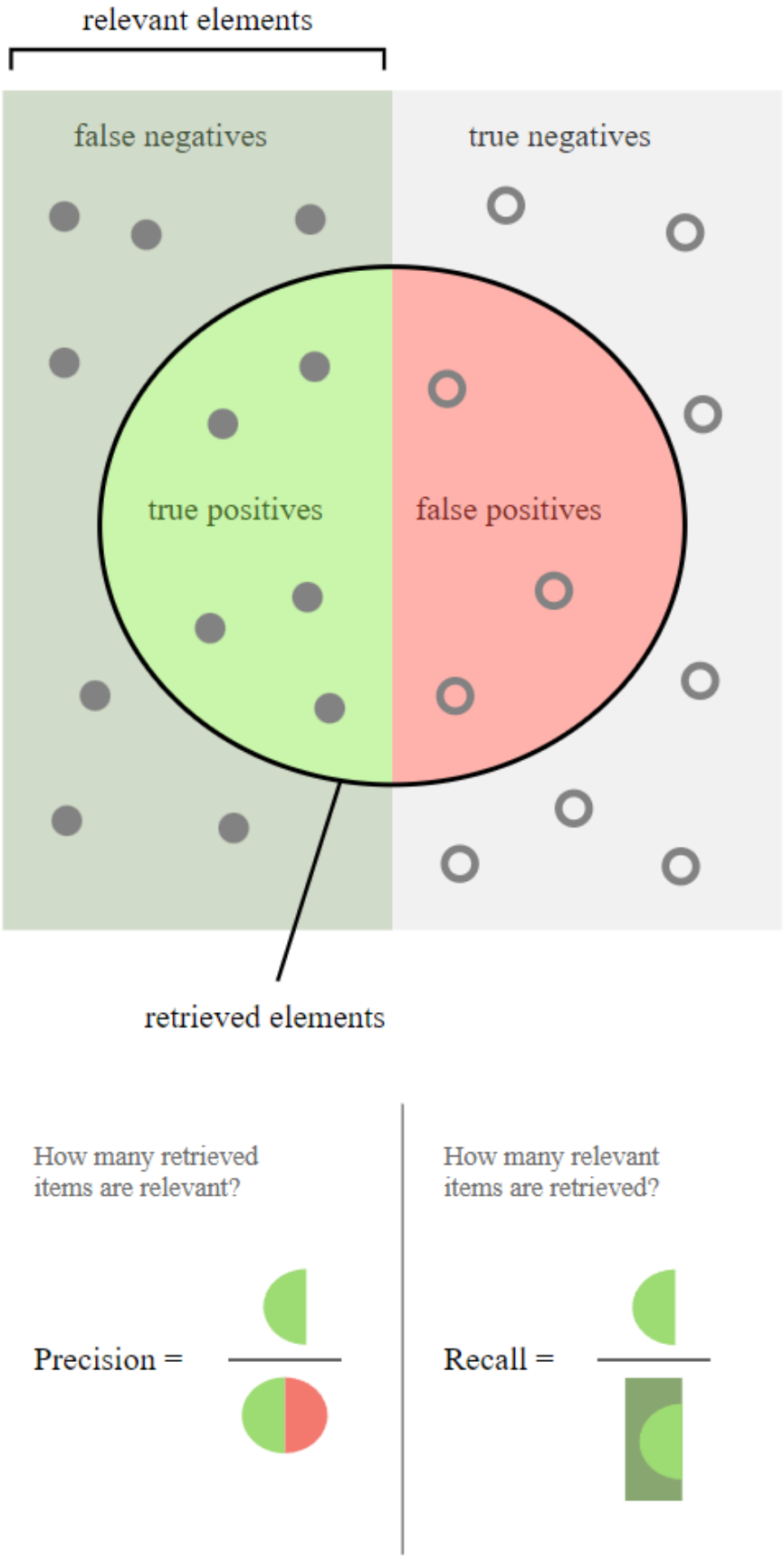

Fig.3: Image for elaborating Precision and Recall.

4. Accuracy:

Accuracy is defined as the ratio of true positive and true negative observation. It can be explained as accuracy will elaborate about how a model will correctly predict the outcome for the number of times it has made predictions.

Accuracy Score
=

$$\frac{\text{(True Positive + True Negative)}}{\text{(True Positive + False Negative + True Negative + False Positive)}} \quad (2021)$$

5. Loss:

The loss function is used to understand and improve due to which the model is facing downfall for accurately predicting the anomalies. (2021)

4 Design Specification:

There are two machine learning models are used in our project for the detection of Malware from portable executables' datasets. Python3 was the base language in which the model was designed for the detection of malware following Jupyter Notebooks. It is a free IDE used for designing and gives output on runtime. Algorithm-related information is discussed below.

1. **Support Vector Machine (SVM)**:

Support Vector Machine (SVM) is a machine learning algorithm opted to find out hyperplane in an N-dimensional space that will distinctly classify the data points.

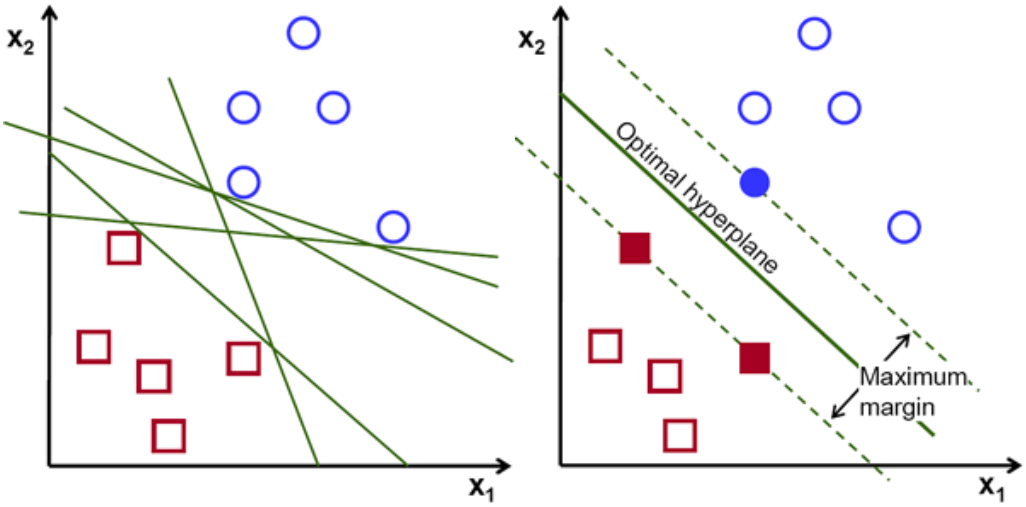

Fig.4: Possible Hyperplanes can be obtained using Support Vector Machine (SVM).

From the figures above it is clear that we can obtain n – number of hyperplanes thereby separating two classes of data points. The objective of experimenting using Support Vector Machine is to obtain the maximum margin that is, the maximum distance between data points of both the classes. (Support Vector Machine — Introduction to Machine Learning Algorithms, 2021)

Hyperplanes and Support Vectors:

With the help of hyperplanes, data points are classified, hyperplanes are the decision boundaries. Data points that fall on another side of the hyperplane can contribute to other classes. As many numbers of features that much is the dimension of the hyperplane. We can face difficulties if the number of features exceeds 3. (Support Vector Machine — Introduction to Machine Learning Algorithms, 2021)

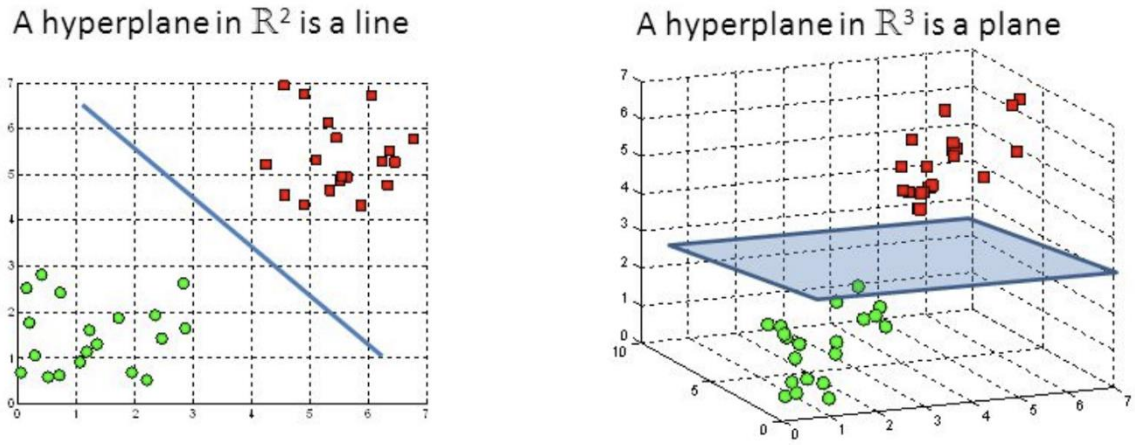

Fig.5: Hyperplanes in 2D and 3D can be obtained using Support Vector Machine (SVM).

Support vectors are the data points that are closer to the hyperplane and due to this the position of the hyperplane is influenced and oriented. With the help of these support vectors, it is possible to maximize the classifier margin. There might change in the position of the hyperplane if support vectors are deleted. (Support Vector Machine — Introduction to Machine Learning Algorithms, 2021)

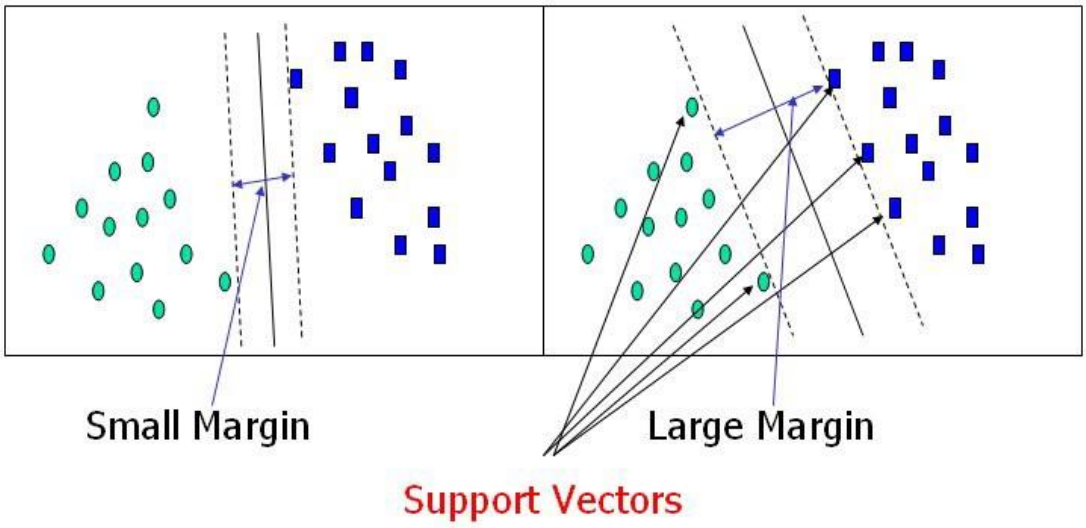

Fig.6: Depicts the size of margin in Support Vector Machine Algorithm (SVM).

2. **Gradient Boosting**:

Gradient Boosting algorithm has originated from Ada Boosting. Gradient boosting trains various models in a gradual, sequential, and additive manner. A major difference between Ada Boosting and Gradient Boosting algorithm is the way these algorithms boost the decision tree. Ada Boosting algorithm identifies weaknesses using high weight data points, however, gradient boosting will identify the weakness using gradients in loss function (*ax + b + e*, where 'e' is a special term specified as '*error term*'). How the model's coefficients are fitting good at underlying data is measured by the loss function. Consider an instance where an obtained dataset is used for predicting prices using regression. In this case, the loss function will measure the true value of houses and predicted prices. The advantage of using the Gradient boosting algorithm is the way it optimizes user-specified cost function rather than how loss function works and does not efficiently correspond to real-world application. (Understanding Gradient Boosting Machines, 2021).

In a gradient boosting algorithm, all trees are connected in series to reduce the errors from previous trees, this is the reason why boosting algorithms are usually slow in learning, however, these algorithms are highly accurate. (An Introduction to Gradient Boosting Decision Trees - Machine Learning Plus, 2021)

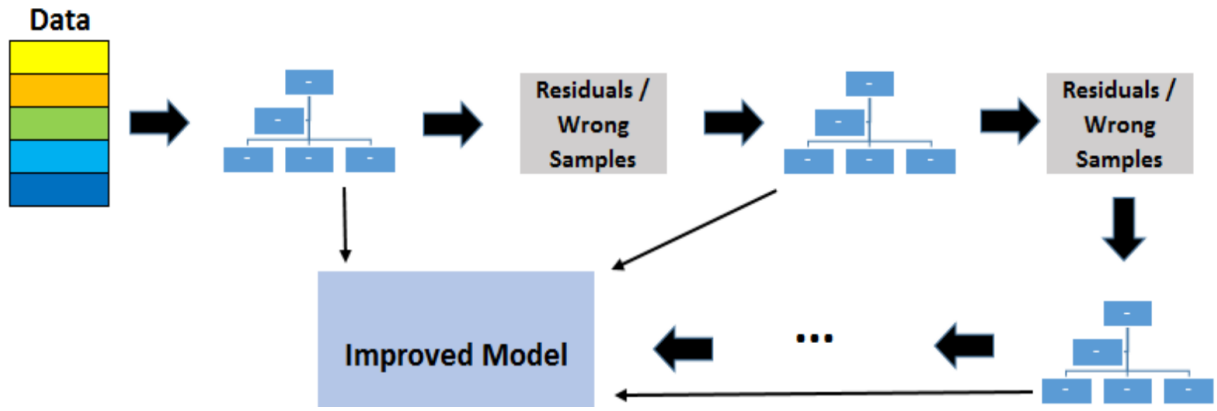

Fig.7: Architecture of Gradient Boosting Algorithm.

Pseudocode:

- Initialize sample weights w (0)i=1l, i=1,…,lwi(0)=1l,i=1,…,l.
- For all t=1,…,Tt=1,…,T
    - Train base algo btbt, let ϵtϵt be its training error.
    - αt=12ln1−ϵtϵtαt=12ln1−ϵtϵt.
    - Update sample weights: w(t)i=w(t−1)ie−αtyibt(xi),i=1,…,lwi(t)=wi(t−1)e−αtyibt(xi),i=1,…,l.
    - Normalize sample weights: w(t)0=∑kj=1w(t)j,w(t)i=w(t)iw(t)0,i=1,…,lw0(t)=∑j=1kwj(t),wi(t)=wi(t)w0(t),i=1,…,l.
- Return ∑Ttαtbt (2021)

5 Implementation:

For the implementation of this project, various software is used. As an IDE Anaconda software was used which included Python and Jupyter notebooks. Also, the latest version of Python 3.97 was downloaded. All the project was implemented on the same machine using Jupyter notebooks and Python was the language used for implementation. Portable Executables consists of various features, a few of the features were extracted. After the features were extracted from the dataset, the dataset was trained initially using machine learning techniques, and later, it was trained to detect malware and benign from the dataset. In a further stage, the accuracy achieved using machine learning techniques was compared and a decision was made about which machine learning technique gave better accuracy for detecting malware. The machine specifications are mentioned below:

Hardware Specifications	
The main memory of the machine	16 Gigabytes
CPU	AMD Ryzen 5 355H
GPU	Nvidia GTX 1650 Super
Hard – Disk	1 Terabyte
Software Specifications	
Operating system	Windows 10
IDE	Anaconda
Programming Language	Python
Designing Software	Jupyter Notebooks
Libraries Used for Implementation	Pandas, NumPy, sklearn, seaborn, matplotlib

Table 1: Specifications' Table.

6 Evaluation:

In this research Portable Executable features were evaluated. Features of Portable Executables were evaluated Optional Header and Section Header. Approximately 19000 samples were evaluated. Various machine learning techniques were used for evaluating the samples, they support Vector Machine (SVM), Gradient Boosting. Every experiment measures the performance of each model. How precise each model is, it is calculated, further recall and F1 – scores are calculated as well. In further go, the accuracy of each model is compared, and based on that decision is made which model is best for detecting the malware. Moreover, in the last stage of the experiment how much was the loss is compared between each model. Each stage is elaborated below:

6.1 Experiment 1 / Precision, Recall, F1 – Score:

6.1.1 Precision:

The chart below elaborates Precision score achieved using Support Vector Machine (SVM) for detecting the malware samples was 0.74 whereas for detecting the benign is 0.00.

	precision	recall	f1-score	support
Benign	0.00	0.00	0.00	1004
Malware	0.74	1.00	0.85	2919
accuracy			0.74	3923
macro avg	0.37	0.50	0.43	3923
weighted avg	0.55	0.74	0.63	3923

Fig.8: Classification report of Support Vector Machine highlighting Precision Score.

	precision	recall	f1-score	support
0	0.83	0.80	0.81	1004
1	0.93	0.94	0.94	2919
accuracy			0.91	3923
macro avg	0.88	0.87	0.87	3923
weighted avg	0.90	0.91	0.91	3923

Fig.9: Classification report of Gradient Boosting highlighting Precision Score.

The chart above elaborates the precision achieved using Gradient Boosting analyzing Malware samples is 0.93 which is very high than the precision achieved using Support Vector Machine (SVM) when malware samples were analyzed. Furthermore, the precision achieved by detecting Benign samples was also very high 0.83 using Gradient Boosting, whereas, using Support Vector Machine (SVM) is very low. In addition to this, the lower is the precision score higher will be the rate of false-positive samples. Thus, from the precision achieved from the Support Vector machine, it is clear that the rate of false-positive observation is high as compared to the precision achieved during the Gradient Boosting experiment.

6.1.2 Recall:

The chart below elaborates recall score achieved by detecting the malware samples using Support Vector Machine (SVM) is 1.00 whereas, for detecting benign samples is 0.00.

	precision	recall	f1-score	support
Benign	0.00	0.00	0.00	1004
Malware	0.74	1.00	0.85	2919
accuracy			0.74	3923
macro avg	0.37	0.50	0.43	3923
weighted avg	0.55	0.74	0.63	3923

Fig.10: Classification report of Support Vector Machine highlighting Recall score

The chart below elaborates the recall score achieved by detecting malware samples using Gradient Boosting is 0.94 which is very high as compared to recall scores achieved using

Support Vector Machine (SVM). Furthermore, recall scores achieved by detecting benign samples was 0.80 which was again high as compared to Support Vector Machine (SVM).

	precision	recall	f1-score	support
0	0.83	0.80	0.81	1004
1	0.93	0.94	0.94	2919
accuracy			0.91	3923
macro avg	0.88	0.87	0.87	3923
weighted avg	0.90	0.91	0.91	3923

Fig.11: Classification report of Gradient Boosting highlighting Recall Score.

6.1.3 F1 – Score:

The chart below elaborates the F1 – Score achieved using the Support Vector Machine (SVM) algorithm when samples were analyzed. After the analysis was done the F1 – Score achieved for Benign samples was 0.00 and the F1 – Score achieved for malware samples was 0.85.

	precision	recall	f1-score	support
Benign	0.00	0.00	0.00	1004
Malware	0.74	1.00	0.85	2919
accuracy			0.74	3923
macro avg	0.37	0.50	0.43	3923
weighted avg	0.55	0.74	0.63	3923

Fig.12: Classification report of Support Vector Machine (SVM) which highlights F1 – Score.

The chart below elaborates the F1 – Score achieved using the Gradient Boosting algorithm when samples were analyzed. After the analysis was done the F1 – Score achieved for Benign samples was 0.81 and the F1 – Score achieved for malware samples was 0.94. The scores achieved during the experiment using the Gradient boosting algorithm are very high as compared to Support Vector Machine (SVM).

	precision	recall	f1-score	support
0	0.83	0.80	0.81	1004
1	0.93	0.94	0.94	2919
accuracy			0.91	3923
macro avg	0.88	0.87	0.87	3923
weighted avg	0.90	0.91	0.91	3923

Fig.13: Classification report of Gradient Boosting which highlights F1 – Score.

6.2 Experiment 2 / Accuracy Comparison:

The chart below elaborates the accuracy achieved while the dataset was analyzed using Support Vector Machine (SVM) for detecting malware, which was 0.74. In other words, it can be said that the accuracy achieved was 74%, which is fair enough.

	precision	recall	f1-score	support
Benign	0.00	0.00	0.00	1004
Malware	0.74	1.00	0.85	2919
accuracy			0.74	3923
macro avg	0.37	0.50	0.43	3923
weighted avg	0.55	0.74	0.63	3923

Fig.14: Classification Report of Support Vector Machine (SVM) highlighting Accuracy.

The chart below elaborates the accuracy achieved while the dataset was analyzed using Gradient Boosting for detecting malware, which was 0.91. In other words, it can be said that the accuracy achieved was 91%, which is very high as compared to the accuracy achieved using Support Vector Machine (SVM). From this, malware detection done using the Gradient Boosting algorithm was better than Support Vector Machine (SVM).

	precision	recall	f1-score	support
0	0.83	0.80	0.81	1004
1	0.93	0.94	0.94	2919
accuracy			0.91	3923
macro avg	0.88	0.87	0.87	3923
weighted avg	0.90	0.91	0.91	3923

Fig.15: Classification Report of Gradient Boosting highlighting Accuracy.

6.3 Discussion:

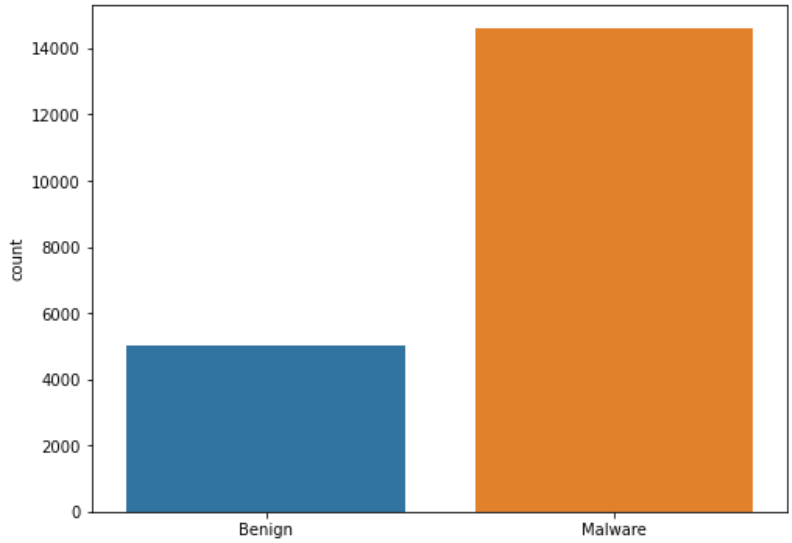

Fig. 16: Number of Malware and Benign Samples Detected.

The histogram above elaborates the count of malware samples and benign samples being detected using the Gradient boosting algorithm. Comparing results of both the algorithms, Support Vector Machine and Gradient Boosting algorithm, the results achieved by using Gradient Boosting algorithm are more accurate. The accuracy for detecting malware by Gradient Boosting algorithm is 91% and accuracy for detecting malware samples using Support Vector is 0.74. Precision value when the experiment was carried out using Support Vector Machine (SVM) for Benign samples is 0.00 and detecting malware samples is 0.73, which is less than Gradient Boosting algorithm. This results that there are more false positives achieved during this experiment. Furthermore, when the Gradient Boosting algorithm was carried out the precision scores achieved for detecting benign samples was 0.83 and for detecting malware samples was 0.93. This makes the picture clear that there are fewer false-positive achieved during experimenting using the Gradient Boosting algorithm as compared to Support Vector Machine (SVM). Confusion metrics of the Gradient Boosting algorithm are shown below, as accuracy produced by Gradient Boosting Algorithm is high as compared to Support Vector Machine (SVM).

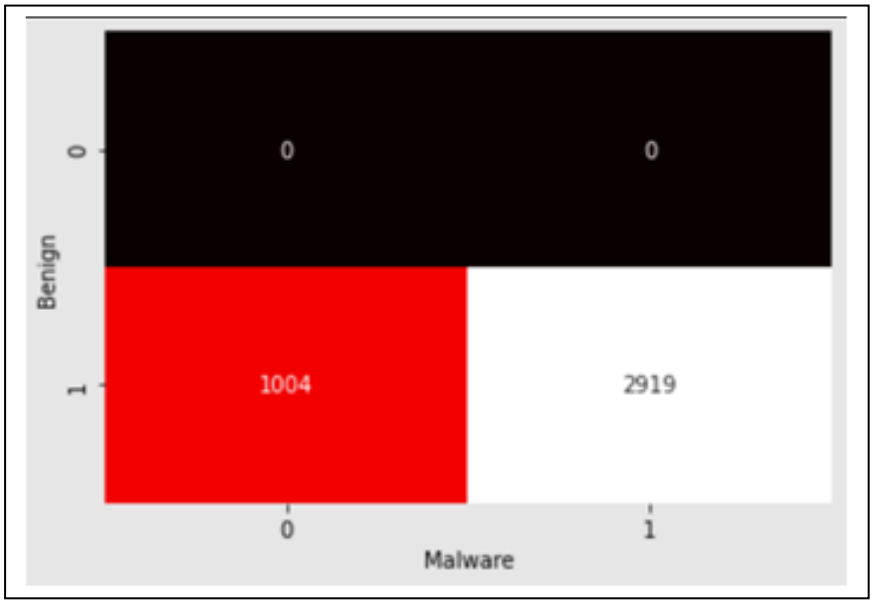

Fig. 17: Confusion Matrix of Gradient Boosting.

7 Conclusion and Future Work:

An experiment was performed using the dataset which had Portable Executable Samples. As the dataset was too complex, its complexity was reduced using Principal Component Analysis (PCA). In the further part of the experiment two models were used for the detection of malware, namely, Support Vector Machine (SVM) and Gradient Boosting. In the next part of the experiment, a dataset was trained and tested using both these models Support Vector Machine (SVM) and Gradient Boosting for detecting the malware. Later, when accuracy from both the models was obtained, Support Vector Machine (SVM) was 74% accurate in the detection of malware and Gradient Boosting 91%. From the accuracies obtained the decision was made and it was concluded that Gradient Boosting gave better accuracy for detection of Malware as compared to Support Vector Machine (SVM). For future work, various other machine learning techniques like Extreme Gradient Boosting, Random Forest Algorithm, Convolution Neural Network, Recurrent Neural Network can be used for the detection of Malware. In addition to it, some online environments can be used for performing the

experiments as running in those environments. As online environments will not use the memory and CPU of the local machine and give better results in less time.

8 References:

Scholarworks.sjsu.edu. 2018. *PE header analysis for Malware Detection*. [online] Available at: <https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1624&context=etd_projects> [Accessed 14 December 2021].

Malware_Detection_System_Based_on_an_Indepth_Analysis_of_the_Portable_Executable_Headers [online] Available at: https://www.researchgate.net/publication/350021076_ [Accessed: 31- Oct- 2021].

Arxiv.org. 2021. Learning the PE Header, Malware Detection with Minimal Domain Knowledge. [online] Available at: <https://arxiv.org/ftp/arxiv/papers/1709/1709.01471.pdf> [Accessed 14 December 2021].

Norma.ncirl.ie. 2021. *Windows Portable Executor Malware detection using Deep learning approaches - NORMA@NCI Library*. [online] Available at: <http://norma.ncirl.ie/4510/> [Accessed 14 December 2021].

Researchgate.net. 2020. Detecting malware evolution using support vector machines [online] Available: https://www.researchgate.net/publication/336611638_Detecting_Malware_Evolution_Using_Support_Vector_Machines [Accessed 14 December 2021].

Researchgate.net. 2010. NewApproach for Detecting Unknown Malicious Executables Available at: https://www.researchgate.net/publication/271103424_NewApproach_for_Detecting_Unknown_Malicious_Executables

Comodo Enterprise. 2021. *What is Malware Analysis? | Different Types of Malware Analysis*. [online] Available at: <https://enterprise.comodo.com/forensic-analysis/malware-analysis-types.php> [Accessed 14 December 2021].

Medium, 2021. Support Vector Machine — Introduction to Machine Learning Algorithms. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed: 29- Nov- 2021].

Medium. 2021. *Understanding Gradient Boosting Machines*. [online] Available at: <https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab> [Accessed 14 December 2021].

Machine Learning Plus. 2021. *An Introduction to Gradient Boosting Decision Trees - Machine Learning Plus*. [online] Available at: <https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees> [Accessed 14 December 2021].

Logix Consulting Managed IT Support Services Seattle. 2021. *What Is Signature-Based Malware Detection?* [online] Available at:

<https://www.logixconsulting.com/2020/12/15/what-is-signature-based-malware-detection> [Accessed 14 December 2021].

Usa.kaspersky.com. 2021. *Cyber Security Resource Center for Threats & Tips | Kaspersky*. [online] Available at: <https://usa.kaspersky.com/resource-center/definitions/heuristic-analysis.> [Accessed 14 December 2021].

Springer. 2021. *Static PE Malware Detection Using Gradient Boosting Decision Trees Algorithm*. [online] Available at: <https://link.springer.com/chapter/10.1007%2F978-3-030-03192-3_17#citeas.> [Accessed 14 December 2021].

Gradient Boosting", *Kaggle.com* 2021. [online] Available at: <https://www.kaggle.com/kashnitsky/topic-10-gradient-boosting.> [Accessed 14 December 2021].

Ds100.org. LogisticRegressionPart2. 2021. [online] Available at: <https://ds100.org/sp20/resources/assets/lectures/lec24/LogisticRegressionPart2.html.> [Accessed 14 December 2021].

DeepAI. 2021. *Long Short-Term Memory*. [online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/long-short-term-memory> [Accessed 14 December 2021].

2021. [online] Available at: <https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/.> [Accessed 14 December 2021].