

MALWARE DETECTION USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for the degree
of Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

Dewesha Sharma (171361)

UNDER THE SUPERVISION OF

Dr. Himanshu Jindal

to

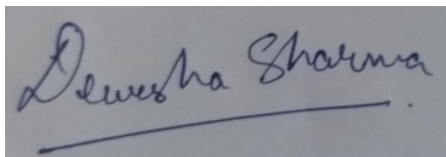


Department of Computer Science Engineering and Information Technology
**Jaypee University of Information Technology, Wagnaghat, Solan -173234,
Himachal Pradesh**

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**Malware Detection using Machine Learning** “ in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering / Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2020 to December 2020 under the supervision of **Dr. Himanshu Jindal** , Assistant Professor (Senior Grade), Department of Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

A photograph of a handwritten signature in blue ink on a light-colored background. The signature reads "Dewesha Sharma" and is underlined with a single horizontal line.

Dewesha Sharma (171361)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

A photograph of a handwritten signature in blue ink on a light-colored background. The signature reads "Himanshu Jindal" and is underlined with two horizontal lines.

Dr. Himanshu Jindal

Assistant Professor (Senior Grade)

Department of Computer Science & Engineering and Information Technology

Dated : 16 May 2021

ACKNOWLEDGEMENT

We wish to express my sense of gratitude towards Dr. Himanshu Jindal, Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat, my guide, for giving me this wonderful opportunity to work with him. We are grateful for his constant encouragement, motivation, cooperation and support which helped me in finishing this project successfully. Without his expert guidance this would not have been possible. We are also grateful to the people whose works we have referred and the details regarding the same are mentioned in the references. We would also like to thank all our friends and lab assistants for extending their help and support at times when it was needed.

TABLE OF CONTENTS

CERTIFICATE	i
ACKNOWLEDGEMENT	ii
TABLE OF CONTENTS	iii
LIST OF ACRONYMS AND ABBREVIATIONS	v
LIST OF FIGURES	vi
LIST OF GRAPHS	viii
LIST OF TABLE	ix
ABSTRACT	x
CHAPTER-1: INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objective	3
1.4 Malware	3
1.4.1 Types of Malware	4
1.4.2 Types of Malware Analysis	8
1.4.3 Types of Malware Detection	9
1.5 Machine Learning	10
1.5.1 Types of Machine Learning	12
1.5.2 Machine Learning algorithms	13
1.5.3 Performance Measures	15
1.6 Organization	17
CHAPTER-2: LITERATURE SURVEY	19
2.1 Related Work	19

CHAPTER-3: SYSTEM DEVELOPMENT	21
3.1 Python	21
3.2 Google Colab	22
3.3 Pandas	22
3.4 NumPy	23
3.5 Scikit - learn	24
3.6 Matplotlib	24
3.7 Dataset	24
3.8 Implementation	25
3.8.1 Preparing the dataset	25
3.8.2 Learning Algorithms	31
CHAPTER-4: PERFORMANCE ANALYSIS	34
4.1 Random Forest	34
4.2 KNN	35
4.3 Gradient Boosting	36
CHAPTER-5: CONCLUSIONS	38
5.1 Conclusions	38
5.2 Future Scope	39
5.3 Applications / Contributions	40
REFERENCES	41

LIST OF ACRONYMS AND ABBREVIATIONS

ML - Machine Learning

AI - Artificial Intelligence

DDoS - Distributed Denial Of Service

Fig. - Figure

Colab - Collaborator

IOT - Internet of things

OS - Operating System

KNN - K nearest neighbor

LIST OF FIGURES

Figure	Page No
1	1
2	2
3	2
4	7
5	11
6	14
7	15
8	17
9	21
10	22
11	22
12	23
13	23
14	24
15	24
16	25
17	26
18	26
19	27
20	27
21	28
22	29

23	29
24	30
25	31
26	31
27	33
28	33
29	34
30	34
31	35
32	35
33	36
34	36
35	36
36	37
37	37

LIST OF GRAPHS

Graph	Page No
1	30
2	32
3	38
4	39

LIST OF TABLE

Table	Page No
1	16
2	20

ABSTRACT

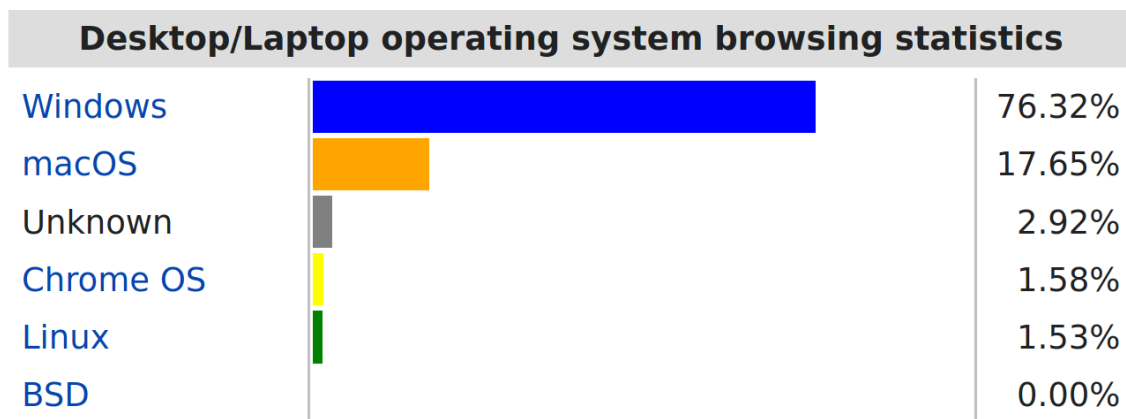
With technology increasing at a fast pace, the digital world is faced by alarming security threats and challenges in the form of malware capable of bringing down organizations and governments. The counter-attacking measures have gotten strong with antivirus companies increasing the signature database which is regularly updated but they are not that efficient and fail in case of polymorphic malware. In this project we present an alternative approach of detecting malicious files by using machine learning algorithms like K-NN, Random Forest and XGBoost and compare their results to determine the best suitable algorithm for our dataset.

CHAPTER 1

INTRODUCTION

1.1 Introduction

It has become impossible now to imagine a world without the internet and with the rate at which technology is growing, the idea of cyborg seems more and more real. Technology is closely integrated with our lives in the form of mobile phones, laptops, smart watches, money transactions etc. Extensive amount of work is done in the field of IoT and a lot of research, time and effort is being devoted into it for building smart houses, smart cities etc. But at the centre of it all is data and technology isn't completely foolproof as it has its own flaws and vulnerabilities. For the scope of this project we are only focusing on laptops and desktops. Out of all the OS available, the majority is still held by windows. According to NetMarketShare, 76.32% users use windows.



Desktop OS market share according to [NetMarketShare](#) for October 2020.^[3] Chrome OS is also based on the [Linux kernel](#).

Fig. 1

The vulnerabilities present in the OS are exploited to gain access to organizations/people's personal data, leaving them compromised and often ends up for sale on dark web or are

extorted in exchange of safely recovering the data. One such recent example is WannaCry which spread through the EternalBlue exploit and ended up affecting almost every country.



Fig.

2

The cases of cyberattacks increased even more during the pandemic. Following image gives us a clearer picture about the same.



Fig. 3

2

Current static and dynamic analysis methods fail to detect malwares efficiently, particularly in case of zero-day attacks or polymorphic malwares. This led us to work towards utilizing machine learning for developing a much better and efficient model which would be able to identify a malicious file and help secure our system in a better way.

1.2 Problem Statement

Considering the rise in the cyber attacks and dynamic nature of technology and malware, a working model capable of detecting malicious files based on certain features needs to be developed. The scope of the model will be limited to only windows executable files.

1.3 Objective

Following objective will be completed upon successful completion of the project -

- A working model capable of detecting malicious files

1.4 Malware

‘Malicious Software’ or more commonly known as ‘Malware’, is a collective name for various malicious softwares like Virus, Ransomware, Spyware, Worms, Trojan horses etc. These malware consist of lines of code and upon execution cause damage and manipulate the data and the system as per the working of the designed malware. These programs can steal, encrypt or delete data, modify or hijack basic computer functions and monitor computer activity without user’s permission.

Earlier malware used to be primitive and the machines were infested through floppy disks but with the advancement in technology, malware have also advanced and polymorphic malware being one such example. They continuously change their signature and are able to evade the signature based detection easily.

1.4.1 Types of Malware

Malware are of many different types and each have their own specific purpose and functionalities. Here we discuss some of them -

● Virus

They are the simplest malware and require human interaction to self replicate. These are capable of damaging the hardware as well as operating system. They are capable of performing following functions -

- Changing the file's size and/or deleting the files
- Formatting the hard disk
- Running down background operations and eventually slowing the system down

● Worms

Computer worms are similar to viruses but the only difference is that they are able to execute on their own without any human interaction. They are able to spread via networks to other computers and replicate themselves in those machines. Worms are of many different types like -

- E-mail Worms
- File Worms shared in the network
- Internet Worms

● Rootkit

Rootkit enables attackers to do privilege escalation and gain administrative access, Hence the name. These are extremely difficult to detect and trace as -

- They can modify and alter the active processes to evade detection by not displaying the rootkit processes.
- They are also able to hide themselves from antivirus by changing their signatures continuously.

● **Ransomware**

Ransomware prevents the user from accessing the computer by encrypting all the data present and the decryption is done only after a ransom is paid to the attacker through some cryptocurrency to avoid any detection from law enforcement agencies. They are able to -

- Format the whole system in case the ransom is not paid
- Stop all the anti-virus, anti-spyware or other security measures causing hindrance in its process

● **Spyware**

As the name suggests, this malware is used to perform espionage i.e spy on the digital activities of the victim. It monitors and gathers information such as personal records, websites visited and the actions taken by them when they visit, their credentials etc via a hidden channel back to the attacker giving them a leverage over the victim. Sometimes these are also used by the organizations in order to target potential consumers for promotional purposes leading to increased number of spams. It is capable of following functions -

- It ends up showing unwanted ads and redirects the victim to desired third party websites.
- It can cause undesirable changes in the system resulting in reduced security.
- It can also act as a gateway to a backdoor giving hackers remote access to the system.

- **Adware**

It can also be considered as a subclass of spyware and the only purpose of it is to display ads and increase traffic to a particular third party site. It also ends up using RAM and additional memory slowing down the system.

- **Trojan**

This malware gets its name from a wooden horse used by Greeks in the Trojan war as just like that wooden horse, a seemingly harmless software or file contains a malicious hidden file and once executed, it enables the attacker to gain unauthorized access to the victim's computer.

- **Backdoor**

As the name suggests, backdoor provides a secret hidden path to the attacker for gaining unauthorized access to the victim's computer. By itself it is completely harmless but provides a medium for various exploits and can also act as a zombie bot used for DDoS attacks.

- **Keylogger**

It logs all the keystrokes made by the user resulting in all the confidential data being compromised resulting in possible cases of blackmail or extortion. One of the advisable methods to protect ourselves from this is by using an onscreen keyboard while entering our credentials.

- **Remote Access Tool**

It helps the attacker to escalate the privileges and provide remote access of the system to the attacker. Following functions can be performed -

- Gaining access to your private data
- Spy on you using your webcam and even pickup all your conversations via microphone
- Block your keyboard or even shutdown or make your system useless
- Install other malware or use it during DDoS attacks

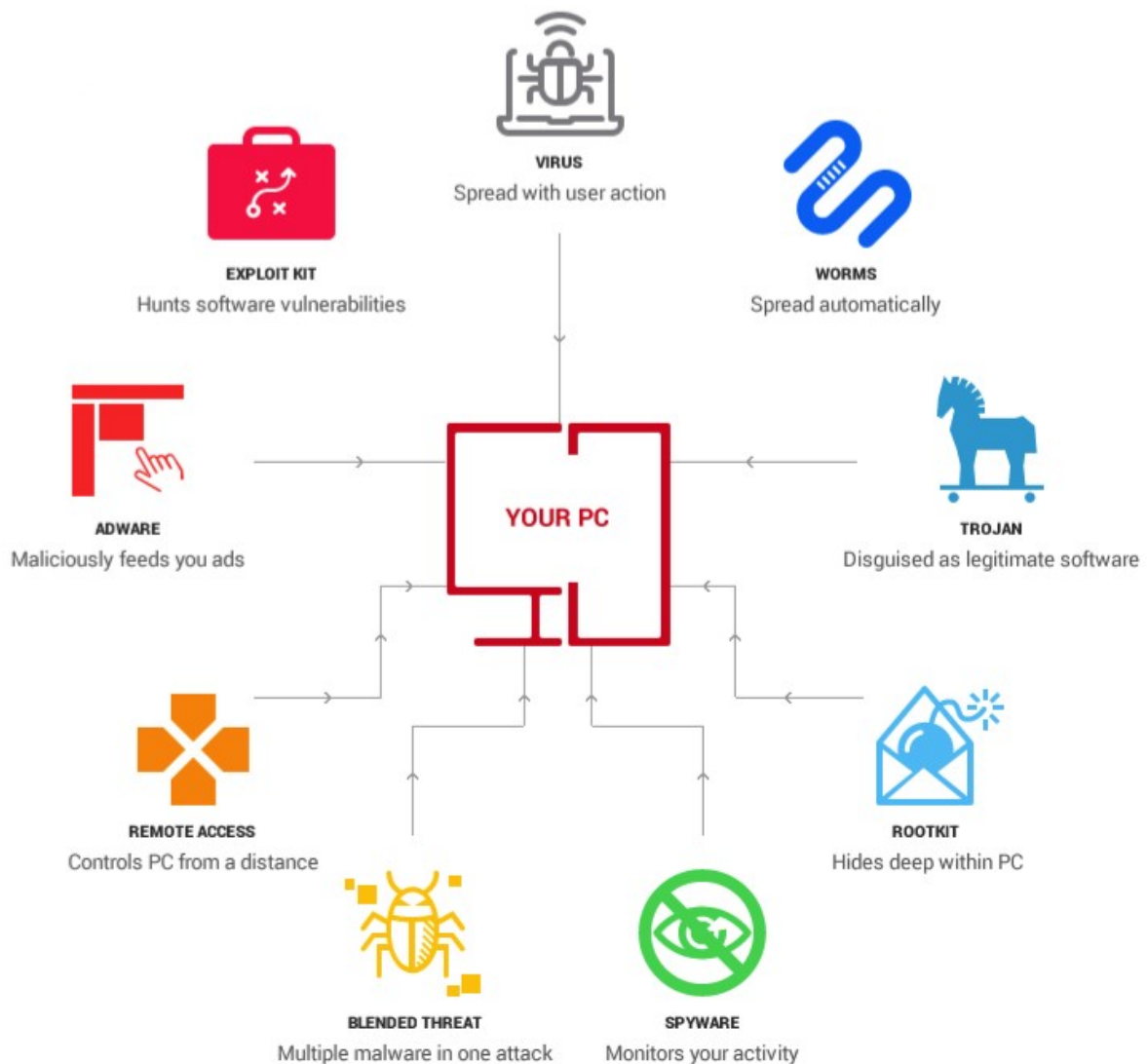


Fig. 4

1.4.2 Types of Malware Analysis

Before we get into malware detection, it is necessary to understand the behavior of the malware. The analysis helps us in observing the behavior and the functionalities of the malware. There are different ways to achieve the outcome with each having its own benefits but the time and knowledge required for each varies greatly. These are as follows -

- **Static Analysis**

Code analysis or also referred to as Static Analysis is achieved by going through the source code of the malware to determine its potential behaviour and properties. This reverse engineering can be achieved by one of the following ways -

- **File Format Inspection**

Metadata is very useful in providing us with many useful information such as file type, date of creation, compile time, functions that have been imported and exported etc.

- **String Extraction**

In string extraction, the output of the code (error messages, status etc) is examined and based on those parameters the working of the malware is inferred.

- **AV Scanning**

The most common way is to do an anti virus inspection on a suspected file and if it is a well known one then all of them might be able to identify it.

- **Disassembly**

Another well known and most reliable method is to run the code in a disassembler which gives us a detailed output about the logics and the working of the program. IDA Pro and Ghidra are some of the most widely used disassemblers.

- **Dynamic Analysis**

Dynamic analysis, also known as behavioural analysis, refers to the process of examining the behaviour of malware in real time. The file is executed in a virtual environment like sandbox or virtual machines and all its activities are monitored and a detailed analysis is done of all the changes and activities done by the file. This type of analysis is much faster than the static analysis.

- **Hybrid Analysis**

As the name suggests, both static and dynamic analysis are done in order to gain a better understanding of the malware. Initially the specific signatures are analyzed and then dynamic analysis is done for getting the overall understanding of the true nature and behaviour of the malware.

1.4.3 Types of Malware Detection

Malware detection methods are implemented for malware detection and prevent them from compromising a system. They are of categorized as follows -

- **Signature Based Detection**

This static method involves the usage of predefined signatures for correctly detecting the malware. Cryptographic hashes such as SHA1 or MD5, file metadata, static strings are some of the signatures. Anti- viruses work on the same model. The file is first analyzed statically by av. In case of a signature match with some other preexisting malicious signature, the file is immediately flagged as infected. This method is useful for well known malware but they fail in case of polymorphic malware as they continuously keep on changing their signature.

● **Behaviour Detection**

Also known as heuristics based analysis, it involves observing the behaviour of a file during execution and flagging it as malicious if found suspicious. Modifying of host files or registry keys may seem harmless but a combination of such activities is definitely a point of concern and any file exceeding a certain threshold raises an alert. The best way to implement this is via virtual environment. Although it will take more time, it is a much safer and foolproof option as compared to signature based detection.

● **Feature Detection**

It can be seen as an application of derivative based detection and is able to overcome the false alarms associated with behaviour detection. This method is similar to identifying the anomalies but different also as it utilizes the characteristics which have been manually developed instead of using ml algorithms.

1.5 Machine Learning

The central idea of machine learning is to build a model that is capable of receiving input data and use its statistical analysis to predict the correct outcome. The model is capable of learning on its own. It can also be seen as a subclass of ai. The basic idea behind this is to train a model to produce some output based on a certain algorithm. A training dataset is provided and the model built on that dataset is used to make predictions.

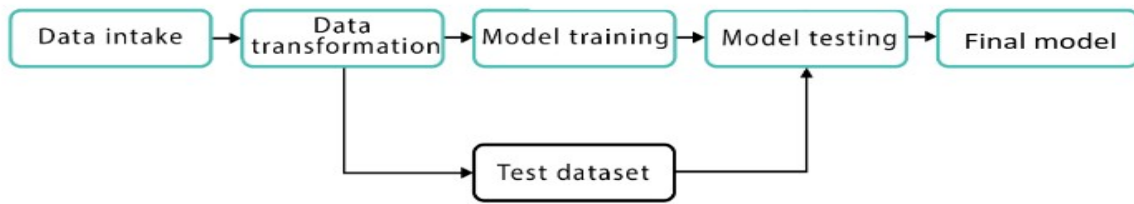


Fig.5

The process consists of 5 stages -

- **Data Intake**

Dataset gets loaded from file and saved in the memory

- **Data Transformation**

Loaded dataset is made suitable for algorithm via transformation and normalization. Data is converted to make it lie in the same range and same format. Feature selection and extraction are performed and data is separated into training set and test set. Training set is used to build our model and the test set is later used for evaluation.

- **Model Training**

Our proposed model is build using the selected algorithm

- **Model Testing**

The built model is trained using our training set and the results are used to built new model which learns from our previous ones.

- **Final Model**

The best model out of all is selected after the required results are achieved or after a certain number of iterations.

1.5.1 Types of Machine Learning

There are two approaches to machine learning and they are as follows -

- **Supervised learning**

In this type of learning, the dataset is mapped to desired outcome and our model gets trained on this dataset where all the factors are known to it. The variables are pre determined and predictions are done based on that variable only.

- **Unsupervised learning**

In unsupervised learning, no predetermined variable is set for our model to train on. These are also known as neural networks and work by collecting and combining the training set with data enabling it to be used for interpreting new data.

- **Semi-Supervised Learning**

The semi-supervised machine learning algorithms uses advantages of both supervised and unsupervised machine learning algorithms for training the dataset and are able to produce highly productive and powerful classifiers.

● **Reinforcement Learning**

Reinforcement learning is a reward based learning in which model directly interacts with the environment and discovers errors.

1.5.2 Machine Learning Algorithms

Following machine learning algorithms have been used by us for our project -

● **K - nearest neighbor**

This is one of the simplest and most accurate learning algorithm. KNN doesn't form any assumptions on its own about the data structure as in real life scenarios theoretical assumptions are rarely obeyed. The model is just like a dataset and no learning is required for the same. KNN is implemented in the following ways -

- Value of k in relation to number of training examples is determined
- Class of each example in test set is determined by calculating the similarities between the test and all the other examples
- First k examples most similar to current example are selected and the class of test example is determined using majority vote
- Information contained in the test file is checked based on the classification
- In case of more test samples go to step 3 and start again
- The quality is calculated using accuracy, true negative rate etc for the current value of k

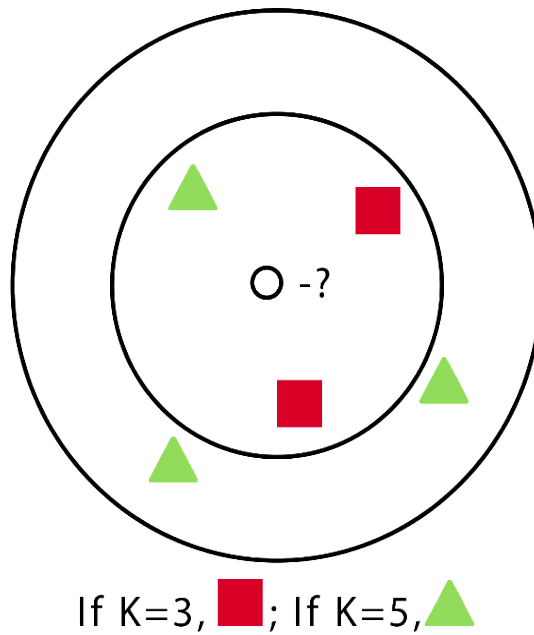


Fig. 6

● Random Forest

It is one of the most popular algorithms as it results in accurate results even without data preparation or modelling. A collection of decision trees results in random forests which results in increased accuracy of prediction. Its algorithms can be described as follows -

- Two third data is chosen randomly and trees are built on it
- Predictor variables are selected randomly and the best split is used to split these nodes
- The number of selected variables is constant for all trees and it is the square root of all predictors
- Rest of data is used for calculating the misclassification rate and is calculated as the overall out-of-bag error rate
- Vote is given by each trained tree to its classification result and the most voted is selected as the result

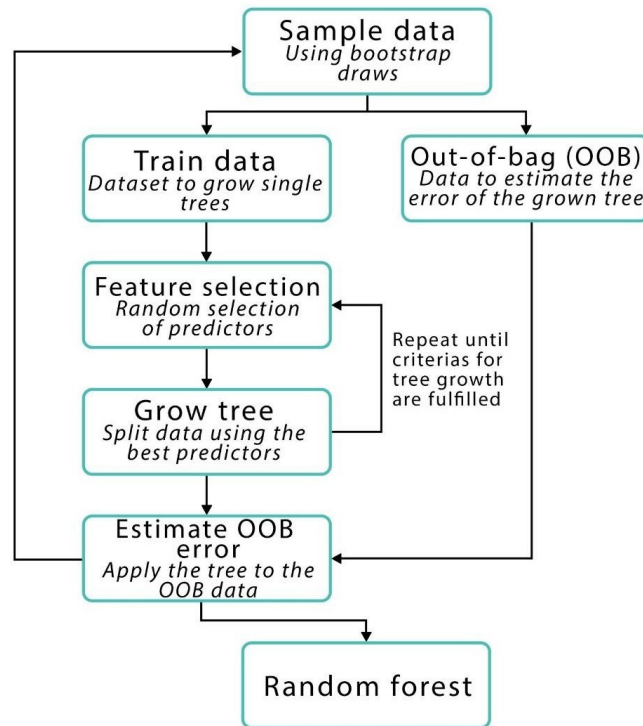


Fig.7

● Gradient Boosting

Gradient boosting is a technique specifically used for regression and classification problems and it produces a model consisting of weak prediction models which are basically decision trees.

1.5.3 Performance Measures

There are various methods to evaluate the performance of the algorithms. One of these methods is to determine the area under the curve or the ROC curve and other parameters which are also known as Confusion Metrics. To evaluate the performance measure of the classification model for a dataset that gives the true values are known, the confusion matrix table is used.

	Predicted Class		
	Class = Yes	Class = No	
Actual Class	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Table 1

The table shown above is known as the confusion matrix and has four sections. The two sections in the green are the True Positive and True Negative and these are the observations which are correctly predicted. The other two sections are in red because these values are wrongly predicted and thus need to be minimized. These sections are false negative and False Positive respectively and occur when there is a contradiction between actual class and the predicted class.

- **True Positives (TP)**

These are the values which are correctly predicted and are positive values which can be described as the positive value of actual class and positive value of predicted class. It is denoted by TP.

- **True Negatives (TN)**

These are the values which are correctly predicted but negative values which refers to the negation of actual class and negation of predicted class. It is denoted by TN.

- **False Positives (FP)**

These are the values which are wrongly predicted but is true in reality i.e. - when we have positive values of actual class but negation in predicted class.

- **False Negative (FN)**

These are the values which are wrongly predicted and negative in actual class.

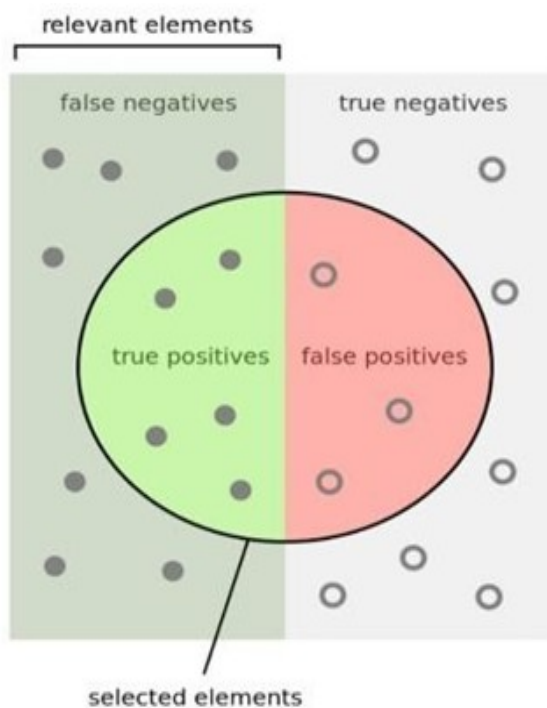


Fig. 8

1.6 Organization

While going through this project you will mainly come across these components -

- **Chapter 1** gives us a basic idea about the project and helps you familiarize with the necessary technical and theoretical aspects of our project
- **Chapter 2** consists of review of other journals and research papers
- **Chapter 3** tells us about the system design and various tools and techniques needed to achieve the same
- **Chapter 4** gives us a comparative study of the performance of each model making it easier for us to choose the best one most suitable for us
- **Chapter 5** provides us with the conclusion and also tell us about the future scope of our project

CHAPTER 2

LITERATURE SURVEY

Literature survey consists of analysing other theoretical works which have been published beforehand. This is a very essential part of the project as it helps us to compare different pieces of literature and studying and analysing them helps us to draw very important inferences. These experiments have been conducted over time by many different experts and analysts and each one of them are unique and different from each other.

These resources are very useful for budding researchers as they can study these and understand the work done in their field upto a given point of time. It can help us by providing us with a specific path we need to work towards.

2.1 Related Work

Some of the works in this field have utilized string or file formats properties for feature representation. The data of PE headers are used for analysing malwares specific to Windows platform. But it isn't the best way to solve the problem as formats of these files can vary drastically(Hung[2]).

Reddy and Pujari [3] have also done breakthrough work in this field. They utilized n-grams for attaining the desired outcome. Byte n-grams are basically overlapping substrings which have been collected in a sliding window fashion. This technology along with the word n-gram and character n-gram are very commonly used in nlp.

This approach has its own drawbacks too. One of the biggest difficulty is that the set of byte strings and the programs is extremely large and classification techniques are unable to be implemented directly.(Reddy and Pujari [3])

In the works done by Kateryna Chumachenko [4], important features were selected and the model was trained using the hashes of well known malware and it yielded a positive result.

Windows API's, hashes and other features were used to analyze it and sandbox testing was also done which lended strength to the results obtained from the model.

Classifier		KNN	SVM	Naive Bayes	J48	Random Forest
		Performance				
Multi-class	Accuracy	87%	87.6%	72.34%	93.3%	95.69%
	Binary	Accuracy	94.6%	94.6%	55%	94.6%
	False-positives	12	20	0	15	9
	False-negatives	8	0	167	5	3
	True-positives	302	310	143	305	307
	True-negatives	49	41	61	46	52

Table 2

CHAPTER 3

SYSTEM DEVELOPMENT

The project was completed in the required time duration in a stepwise manner. Following timeline gives us a better understanding of the stages in which our project was divided and the time taken to complete that stage of development.

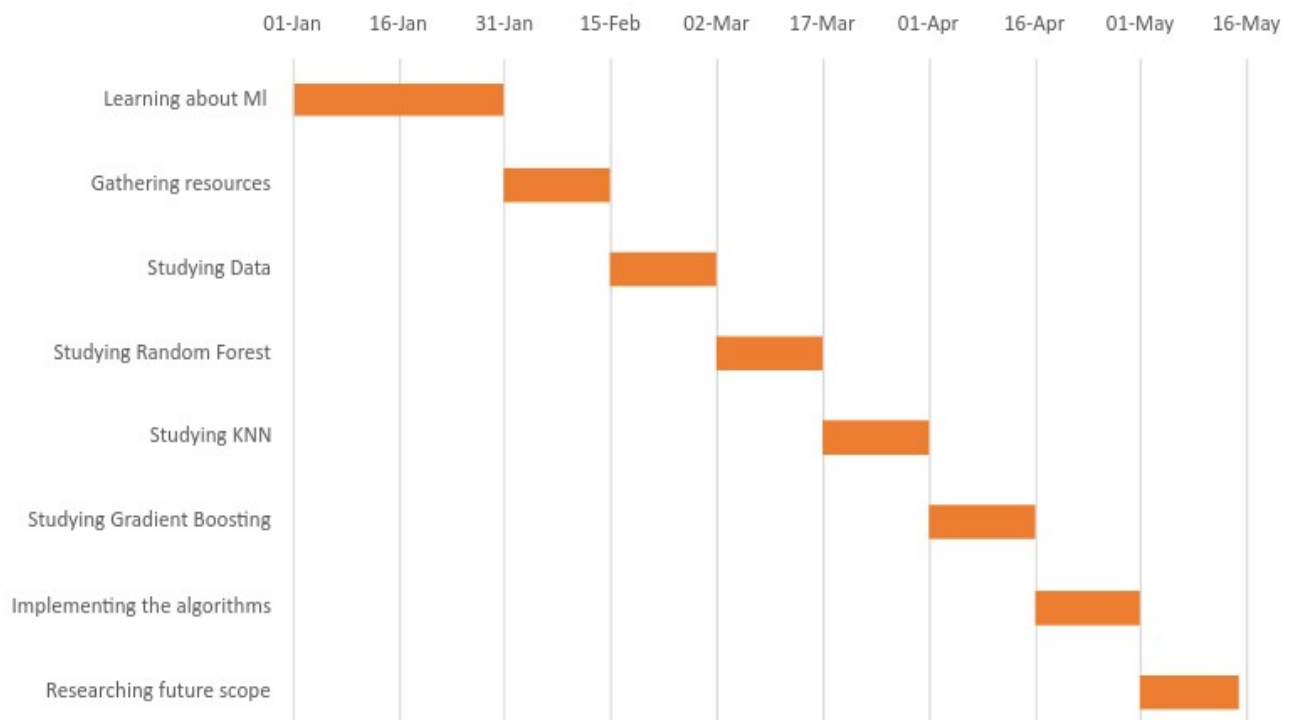


Fig. 9

We have used many Tools and technologies used for our project and all are discussed here -

3.1 Python

- Python is a high level, general purpose programming language developed in 1991 by Guido van Rossum.

- It is extremely favourable because of its code readability. It is an object oriented language having a large number of libraries and modules making it one of the go to languages in various fields.
- It has important libraries like Pandas and Numpy which are extensively used in the field of deep learning, machine learning and ai as they help in visualizing the data.



Fig. 10

3.2 Google Colab

- Google Colab is a powerful software made by Google Inc which is used by many Data Scientists for visualization of Data as well as preprocessing the data.
- It provides a free software platform where various Machine Learning algorithms can be implemented.
- Free Access to Industry Grade GPU's and Cpu processing is provided by Google .



Fig. 11

3.3 Pandas

- This happens to be the most sought after tool for data scientist
- This python library is useful in data analysis and manipulation

- It helps in manipulation, transformation, cleaning and analysis of data



Fig. 12

3.4 Numpy

- It is python library useful for numerical analysis
- The library consists of matrix ds and multidimensional arrays
- Also has functions capable of performing linear algebra, fourier transformation and matrices



Fig. 13

3.5 Scikit - learn

- It is a free ml python library and one of the most useful one
- It contains many tools useful in modeling such as classification, clustering, regression etc.
- It provides many learning algorithms through a consistent interface.



Fig. 14

3.6 Matplotlib

- It is a python library used for plotting of graphs or other visuals.
- These visuals can be static, animated or even interactive

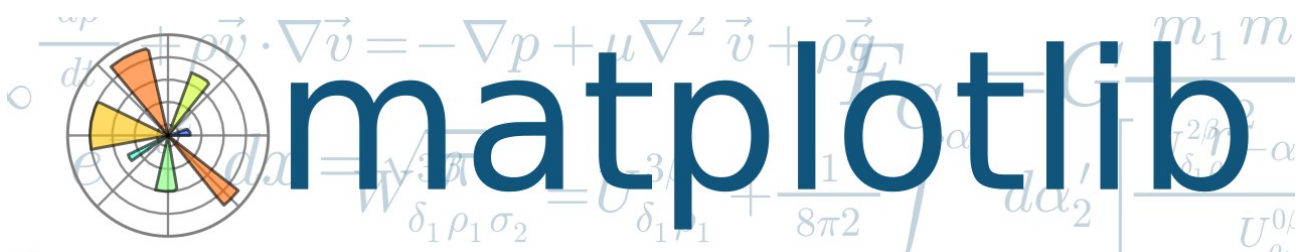


Fig. 15

3.7 Dataset

In order to train our model, we selected a dataset curated by a security blogger Prateek Lalwani. The dataset consists of features which are extracted from the following sources

- 41,323 Windows binaries (.exe and .dlls) as legitimate files
- 96,724 malware files which are downloaded from VirusShare website

In total, our dataset consists of 1,38,048 lines. As is visible from the pie chart below the ratio of legitimate to malicious is approximately 1:3.

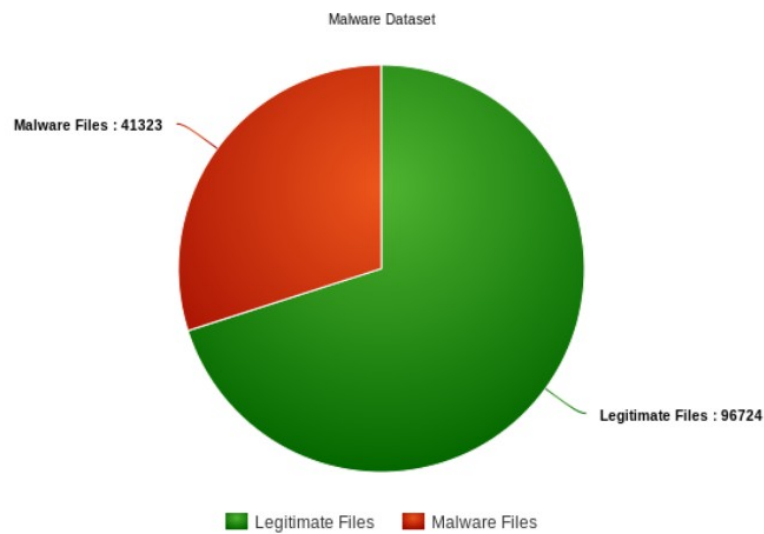


Fig. 16

3.8 Implementation

As per our ongoing discussion, some of the security threats and challenges faced by the digital world have been discussed in depth. Keeping them in mind we will go with a static analysis method for achieving the desired outcome.

3.8.1 Preparing the dataset

● Importing Dataset

Since we are working on colab , the first steps towards processing our data and implementing the model would be to import the dataset.

```
from google.colab import files

uploaded = files.upload()

Choose files MalData.csv
• MalData.csv(text/csv) - 50668563 bytes, last modified: 12/05/2021 - 100% done
Saving MalData.csv to MalData.csv
```

Fig. 17

● Extracting features

Our dataset in total consists of 56 features.

```
Data columns (total 57 columns):
# Column
---
0 Name
1 md5
2 Machine
3 SizeOfOptionalHeader
4 Characteristics
5 MajorLinkerVersion
6 MinorLinkerVersion
7 SizeOfCode
8 SizeOfInitializedData
9 SizeOfUninitializedData
10 AddressOfEntryPoint
11 BaseOfCode
12 BaseOfData
13 ImageBase
14 SectionAlignment
15 FileAlignment
16 MajorOperatingSystemVersion
17 MinorOperatingSystemVersion
18 MajorImageVersion
19 MinorImageVersion
20 MajorSubsystemVersion
21 MinorSubsystemVersion
22 SizeOfImage
23 SizeOfHeaders
24 CheckSum
25 Subsystem
26 DllCharacteristics
27 SizeOfStackReserve
28 SizeOfStackCommit
29 SizeOfHeapReserve
30 SizeOfHeapCommit
31 LoaderFlags
32 NumberOfRvaAndSizes
33 SectionsNb
```

Fig. 18

```

34 SectionsMeanEntropy
35 SectionsMinEntropy
36 SectionsMaxEntropy
37 SectionsMeanRawsize
38 SectionsMinRawsize
39 SectionMaxRawsize
40 SectionsMeanVirtualsize
41 SectionsMinVirtualsize
42 SectionMaxVirtualsize
43 ImportsNbDLL
44 ImportsNb
45 ImportsNbOrdinal
46 ExportNb
47 ResourcesNb
48 ResourcesMeanEntropy
49 ResourcesMinEntropy
50 ResourcesMaxEntropy
51 ResourcesMeanSize
52 ResourcesMinSize
53 ResourcesMaxSize
54 LoadConfigurationSize
55 VersionInformationSize
56 legitimate

```

Fig. 19

As mentioned above our total dataset consists of 1,38,048 lines and the first 41,323 lines consists of legitimate files and their features whereas the rest of the dataset contains malicious files. Legitimate files have legitimate value as '1' whereas malicious files have legitimate value '0'.

```

2 Name md5 Machine SizeOfOptionalHeader \
2 setup.exe 4d92f518527353c0db88a70fddcfd390 332 224
2 Characteristics MajorLinkerVersion MinorLinkerVersion SizeOfCode \
2 3330 9 0 517120
2 SizeOfInitializedData SizeOfUninitializedData AddressOfEntryPoint \
2 621568 0 350896
2 BaseOfCode BaseOfData ImageBase SectionAlignment FileAlignment \
2 4096 811008 771751936.0 4096 512
2 MajorOperatingSystemVersion MinorOperatingSystemVersion \
2 5 1
2 MajorImageVersion MinorImageVersion MajorSubsystemVersion \
2 0 0 5
2 MinorSubsystemVersion SizeOfImage SizeOfHeaders CheckSum Subsystem \
2 1 1150976 1024 1159817 2
2 DllCharacteristics SizeOfStackReserve SizeOfStackCommit \
2 32832 1048576 4096
2 SizeOfHeapReserve SizeOfHeapCommit LoaderFlags NumberOfRvaAndSizes \
2 1048576 4096 0 16
2 SectionsNb SectionsMeanEntropy SectionsMinEntropy SectionsMaxEntropy \
2 4 6.409558 4.885191 7.600957
2 SectionsMeanRawsize SectionsMinRawsize SectionMaxRawsize \
2 273408.0 21504 517120
2 SectionsMeanVirtualsize SectionsMinVirtualsize SectionMaxVirtualsize \
2 284498.0 21456 516760
2 ImportsNbDLL ImportsNb ImportsNbOrdinal ExportNb ResourcesNb \
2 14 235 21 1 11
2 ResourcesMeanEntropy ResourcesMinEntropy ResourcesMaxEntropy \
2 4.426324 2.846449 5.271813
2 ResourcesMeanSize ResourcesMinSize ResourcesMaxSize \
2 31102.272727 104 270376
2 LoadConfigurationSize VersionInformationSize legitimate
2 72 18 1

```

Fig. 20

In the above figure we can clearly see that the legitimate value is 1 which implies that the data is of a legitimate file.

● Organizing dataset

After importing the data and extracting the features, we organize the dataset into legit and mal sets.

```
maldata = pd.read_csv("MalData.csv", sep="|")
legit = maldata[0:41323].drop(["legitimate"], axis=1)
mal = maldata[41323:].drop(["legitimate"], axis=1)

print("The legit dataset is %s samples and %s features"%(legit.shape[0],legit.shape[1]))
print("The malware dataset is %s samples and %s features"%(mal.shape[0],mal.shape[1]))

The legit dataset is 41323 samples and 56 features
The malware dataset is 96724 samples and 56 features
```

Fig. 21

● Feature Selection

After organizing and dividing the dataset, we move towards selecting the most important features from our dataset. The dataset consists of 56 features but not all will be of that much importance. So we use tree based feature selection to assign weight to features and select the most important ones out from the 56 features.

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
```

Fig. 22

We import the necessary libraries and proceed towards selecting the features.

```
data_in = maldata.drop(['Name', 'md5', 'legitimate'], axis =1).values
labels = maldata['legitimate'].values
feslec = ExtraTreesClassifier().fit(data_in,labels)
select = SelectFromModel(feslec, prefit = True)
data_in_new = select.transform(data_in)

print(data_in.shape, data_in_new.shape)

(138047, 54) (138047, 14)
```

Fig. 23

From the above output image we can see that out of the total 54 features (removing the name, md5 and legitimate column as they are not necessary in our scenario) only 14 were important and selected using the tree based feature selection. We can also see the features selected and the weight assigned to each one of them.

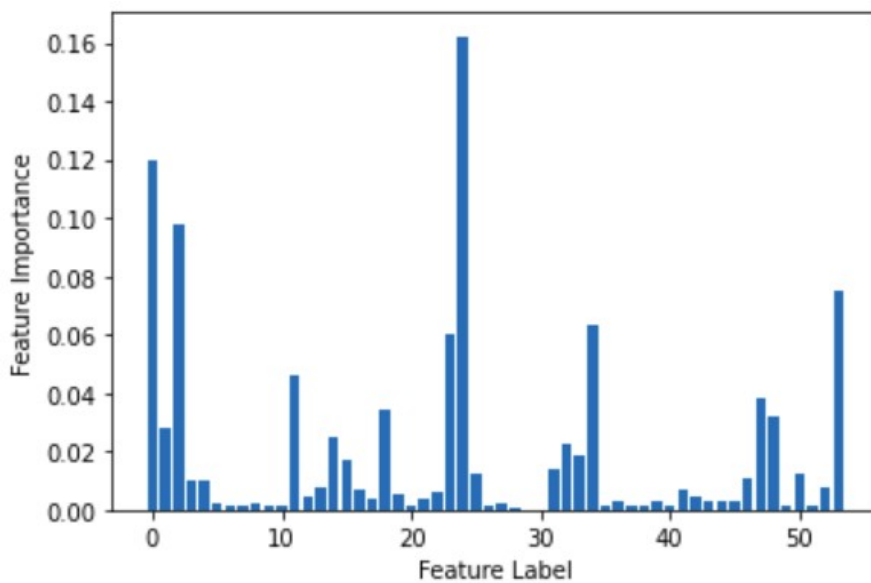

```

1 DllCharacteristics 0.16226741327135288
2 Machine 0.11935542710537982
3 Characteristics 0.0978731895331685
4 VersionInformationSize 0.07488837624889287
5 SectionsMaxEntropy 0.06293680781490484
6 Subsystem 0.059796529154792596
7 ImageBase 0.04579979836337387
8 ResourcesMinEntropy 0.03832223451688429
9 MajorSubsystemVersion 0.03435805162432689
10 ResourcesMaxEntropy 0.03214574948962828
11 SizeOfOptionalHeader 0.027777444767734083
12 MajorOperatingSystemVersion 0.024546137842613103
13 SectionsMeanEntropy 0.02255286537499676
14 SectionsMinEntropy 0.018657298140640655

```

Fig. 24

We get the list of features selected along with their weightage. ‘DllCharacteristics’ having the highest weightage of 0.16 to ‘SectionsMinEntropy’ having the lowest weightage out of all the selected features i.e 0.018. We can visualize this as a graph also.



Graph 1

- **Splitting the dataset**

After feature selection we move towards splitting the dataset into training and testing sets. We can divide the dataset into any ration but here we go with the 80:20 ratio i.e. 80% training size and 20% test size.

```
legit_train, legit_test, mal_train, mal_test = train_test_split(data_in_new, labels, test_size = 0.2)
```

Fig. 25

3.8.2 Learning Algorithms

In this we discuss the algorithms that have to be implemented.

- **Random Forest**

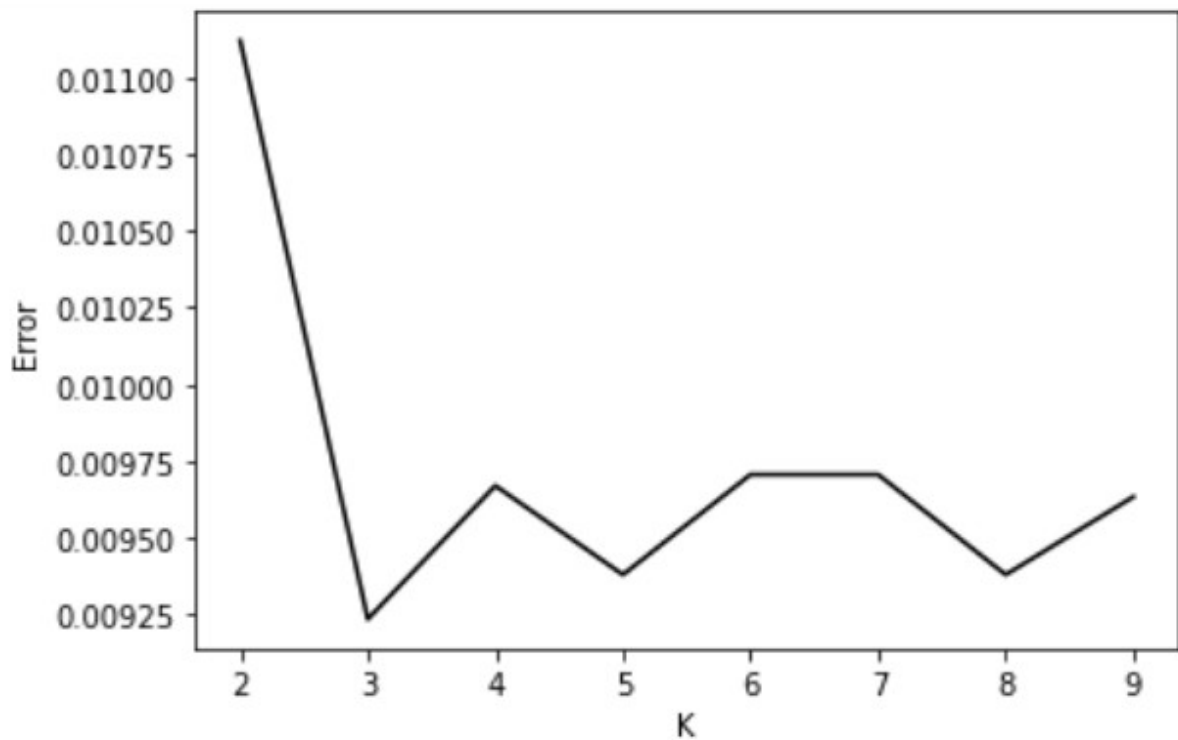
First algorithm we are going to implement is Random Forest. In order to successfully implement this algorithm it is necessary to clearly specify the estimators. For our project we have set the value of the estimator to 50.

```
from sklearn.ensemble import RandomForestClassifier
legit_train, legit_test, mal_train, mal_test = train_test_split(data_in_new, labels, test_size = 0.2)
classif = RandomForestClassifier(n_estimators=50)
classif.fit (legit_train,mal_train)
rf = classif.score(legit_test,mal_test)*100
```

Fig. 26

- **KNN**

The next algorithm we are going to implement is knn. In order to implement this algorithm, we first need to identify the value of k or the nearest neighbours that will be considered for identifying the file. In order to obtain the most optimum value of k, we use the elbow curve method.



Graph 2

We check the error percentage for each value of k ranging from 1 to 10. See and the minimum error value is at k = 3. So we implement the knn algorithm using the value 3.

```
knn = KNeighborsClassifier(n_neighbors=3, p=2)
knn.fit(legit_train,mal_train)

k =knn.score(legit_test,mal_test)*100
```

Fig. 27

● Gradient Boosting

In order to implement the gradient boosting algorithm, we need to specify the estimator. So we select the estimator value as 50 and implement the gradient boosting algorithm.

```
from sklearn.ensemble import GradientBoostingClassifier
grad_boost = GradientBoostingClassifier(n_estimators =50)
grad_boost.fit(legit_train,mal_train)
gb=grad_boost.score(legit_test,mal_test)*100
```

Fig. 28

CHAPTER 4

PERFORMANCE ANALYSIS

In order to compare and analyze the performance of our algorithms we take the use of confusion matrix and also take into consideration the accuracy of each algorithm.

4.1 Random Forest

Random forest was successfully implemented, so we look at the confusion matrix of the same.

```
from sklearn.metrics import confusion_matrix

result = clf.predict(legit_test)
conf_mat = confusion_matrix(mal_test, result)

print(conf_mat)

[[19305   84]
 [   52 8169]]
```

Fig. 29

We get the above confusion matrix for random forest algorithm and as is visible from it, the false positives and false negatives are very low or negligible as compared to true positives or true negatives.

```
print("False Positives: ", conf_mat[0][1]/sum(conf_mat[0])*100)
print("False Negatives: ", conf_mat[1][0]/sum(conf_mat[1])*100)

False Positives: 0.4332353396255609
False Negatives: 0.6325264566354458
```

Fig. 30

```
print(" The accuracy is: ",rf)
The accuracy is: 99.45671858022456
```

Fig. 31

Based on the above confusion matrix, we get an accuracy of 99.46 % for the random forest algorithm.

4.2 KNN

For knn algorithm the confusion matrix is as follows -

```
result1 = knn.predict(legit_test)
conf_mat1 = confusion_matrix(mal_test,result1)

print(conf_mat1)

[[19223  166]
 [   95 8126]]
```

Fig. 32

As we can see the number of false positives and false negatives has increased as compared to the random forest algorithm. On calculating the percentage of false negatives and false positives we get the following value -

```
print("False Positives: ", conf_mat1[0][1]/sum(conf_mat1[0])*100)
print("False Negatives: ", conf_mat1[1][0]/sum(conf_mat1[1])*100)

False Positives: 0.8561555521171799
False Negatives: 1.15557718039168
```

Fig. 33

Based on the above confusion matrix, we get the following accuracy value for knn algorithm

```
print("The accuracy is: ",k)

The accuracy is: 99.06917783411807
```

Fig. 34

4.3 Gradient Boosting

In order to verify the accuracy of gradient boosting algorithm we first take a look at its confusion matrix.

```
result2 = grad_boost.predict(legit_test)
conf_mat2 = confusion_matrix(mal_test,result2)

print(conf_mat2)

[[19234  155]
 [ 157 8064]]
```

Fig. 35

Based on the above confusion matrix if we calculate the percentage of false negatives and false positives their value is still low but high as compared to other two algorithms.

```
print("False Positives: ", conf_mat2[0][1]/sum(conf_mat1[0])*100)
print("False Negatives: ", conf_mat2[1][0]/sum(conf_mat1[1])*100)

False Positives: 0.7994223528804992
False Negatives: 1.9097433402262498
```

Fig. 36

```
print("The accuracy is: ",gb)

The accuracy is: 98.75407461064832
```

Fig. 37

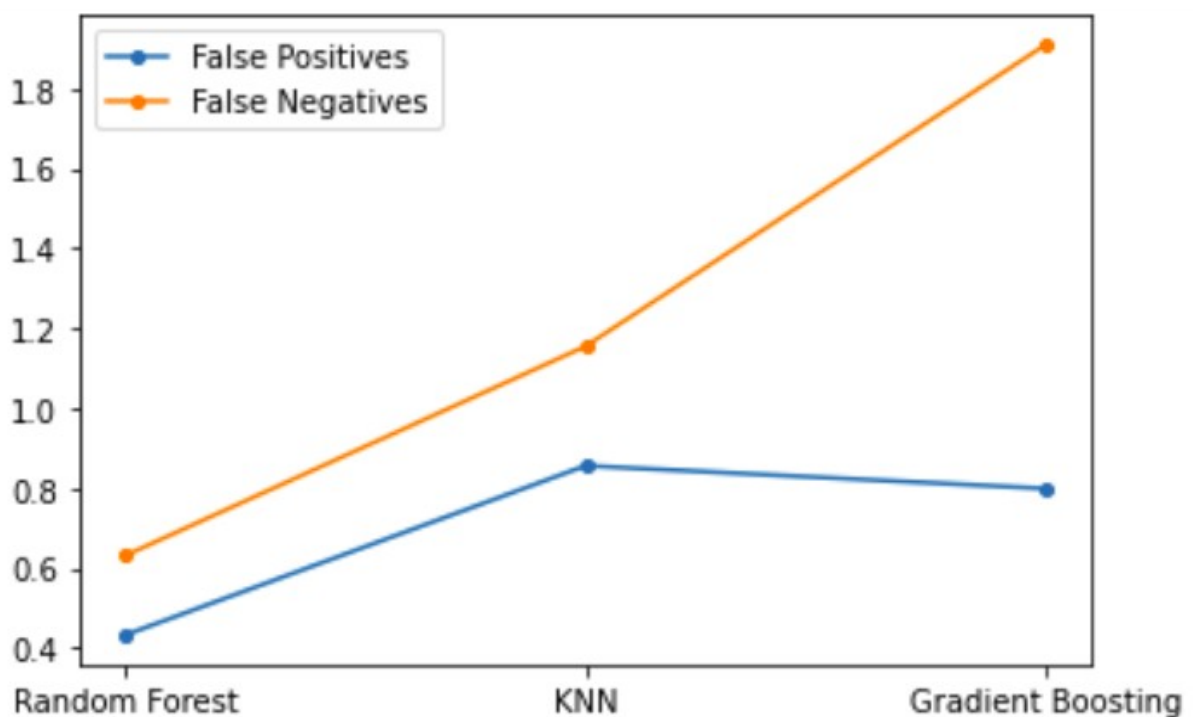
Based on the above confusion matrix, we achieve the accuracy of 98.75% .

CHAPTER 5

CONCLUSIONS

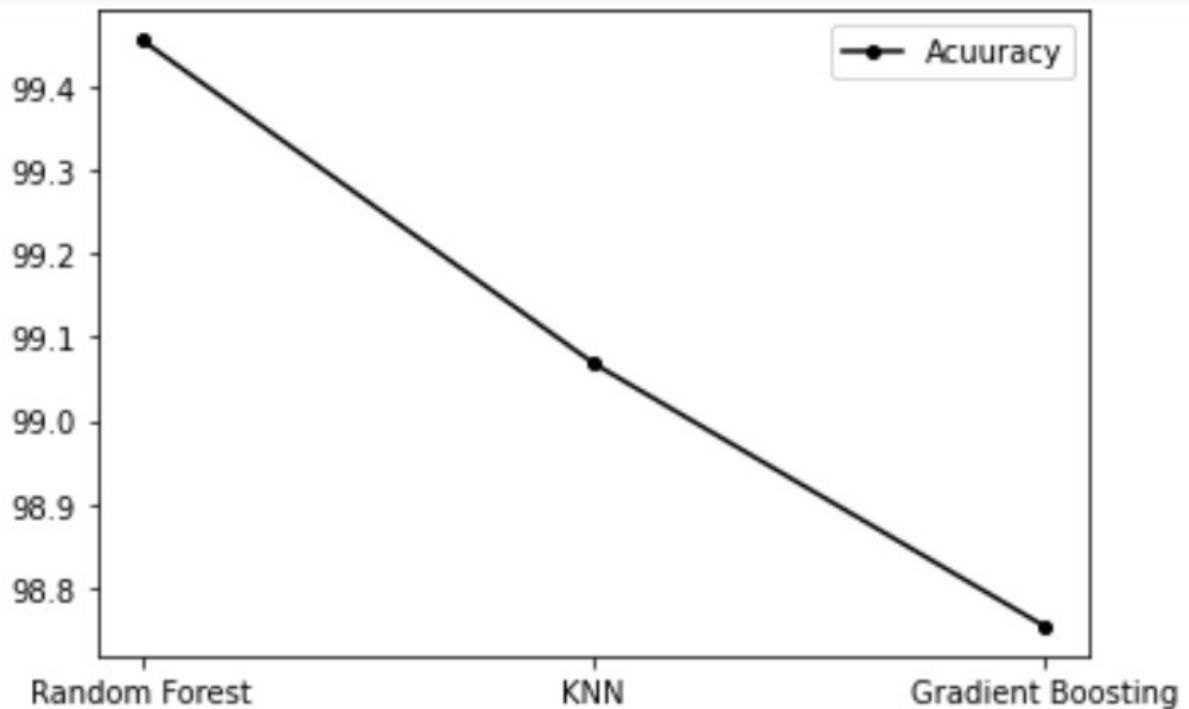
5.1 Conclusions

We plot a graph of false positives and false negatives for better visualization of the results.



Graph 3

Based on the graph we can conclude that random forest will have higher accuracy as compared to the other two algorithms. After successfully implementing all the algorithms we can infer that the highest accuracy of 99.45% was achieved by random forest followed by knn achieving an accuracy of 99.06%. Gradient boosting comes at last with an accuracy of 98.75%. The result can be conceptualized better as a graph.



Graph 4

Hence the best algorithm out of all the algorithms implemented for our project is random forest with an accuracy of 99.45%.

5.2 Future Scope

Considering the fast changing pace of technology a lot can be done to further improve upon our proposed model

- The Algorithms implemented are sufficient by themselves but nowadays Neural Networks play an important role in classification problems.
- Neural Networks can be implemented instead of implementing machine learning algorithms which are much better in terms of unsupervised learning.
- Specific feature selection can also be done in order to remove false positives

5.3 Applications

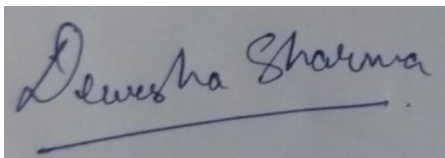
- In the field of security this has wide and far reaching implications as it can be used to detect malicious files and many av companies are working towards incorporating it in their software.

REFERENCES

1. Firdausi, I., Erwin, A. and Nugroho, A.S., 2010, December. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 second international conference on advances in computing, control, and telecommunication technologies* (pp. 201-203). IEEE.
2. Gavriluț, D., Cimpoeșu, M., Anton, D. and Ciortuz, L., 2009, October. Malware detection using machine learning. In *2009 International Multiconference on Computer Science and Information Technology* (pp. 735-741). IEEE.
3. Amos, B., Turner, H. and White, J., 2013, July. Applying machine learning classifiers to dynamic android malware detection at scale. In *2013 9th international wireless communications and mobile computing conference (IWCMC)* (pp. 1666-1671). IEEE.
4. Narudin, F.A., Feizollah, A., Anuar, N.B. and Gani, A., 2016. Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, 20(1), pp.343-357.
5. Ahmed, F., Hameed, H., Shafiq, M.Z. and Farooq, M., 2009, November. Using spatio-temporal information in API calls with machine learning algorithms for malware detection. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence* (pp. 55-62).
6. Santos, I., Devesa, J., Brezo, F., Nieves, J. and Bringas, P.G., 2013. Opem: A static-dynamic approach for machine-learning-based malware detection. In *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions* (pp. 271-280). Springer, Berlin, Heidelberg.
7. Ham, H.S. and Choi, M.J., 2013, October. Analysis of malware detection performance using machine learning classifiers. In *2013 international conference on ICT Convergence (ICTC)* (pp. 490-495). IEEE.
8. Reddy, Krishna Sandeep, and Arun Pujari. 2006. N-gram analysis for computer virus detection.

9. Hung, Pham Van. 2011. An approach to fast malware classification with machine learning techniques.

Dewesha_Malware_Detection.d OCX by

A rectangular box containing a handwritten signature in blue ink that reads "Dewesha Sharma". A horizontal line is drawn below the signature.

Dewesha Sharma (171361)

Submission date: 15-May-2021 11:50PM (UTC+0530)

Submission ID: 1586749405

File name: Dewesha_Malware_Detection.docx (2.4M)

Word count: 5709

Character count: 28491

MALWARE DETECTION USING MACHINE LEARNING

Project report submitted in partial fulfillment of the requirement for the degree
of Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

By

Dewesha Sharma (171361)

UNDER THE SUPERVISION OF

Dr. Himanshu Jindal

to



Department of Computer Science Engineering and Information Technology,
Jaypee University of Information Technology, Wanknaghat, Solan -173234,
Himachal Pradesh

■

■

■

Candidate's Declaration

I hereby declare that the work presented in this report entitled "**Malware Detection using Machine Learning**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering / Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2020 to December 2020 under the supervision of Dr. Himanshu **Jindal**, Assistant Professor (Senior Grade), Department of Computer Science & Engineering and Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Dewesha Sharma

171361

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Himanshu Jindal

Assistant Professor (Senior Grade)

Department of Computer Science & Engineering and Information Technology

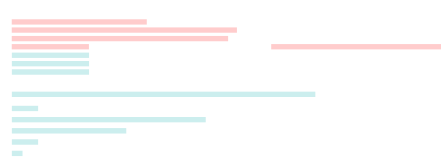
Dated :

ACKNOWLEDGEMENT

We wish to express my sense of gratitude towards Dr. Ravindra Bhatt, Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat, my guide, for giving me this wonderful opportunity to work with him. We are grateful for his constant encouragement, motivation, cooperation and support which helped me to finish this project successfully. Without his expert guidance this would not have been possible. We are also grateful to the people whose works we have referred and the details regarding the same are mentioned in the references. We would also like to thank all our friends and lab assistants for extending their help and support at times when it was needed.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF ACRONYMS AND ABBREVIATIONS	v
LIST OF FIGURES	vi
LIST OF TABLES	vii
PREFACE	viii
CHAPTER 1: INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Motivation	3
1.4.1 Types of Malware	4
1.4.2 Types of Malware Analysis	8
1.4.3 Types of Malware Detection	9
1.5 Machine Learning	10
1.5.1 Types of Machine Learning	12
1.5.2 Machine Learning algorithms	13
1.5.3 Performance Metrics	15
1.6 Organization	17
CHAPTER 2: LITERATURE SURVEY	19
2.1 Related Work	19
CHAPTER 3: SYSTEM DEVELOPMENT	23



1.1 Introduction	1.1
1.2 Objectives of the Study	1.2
1.3 Scope of the Study	1.3
1.4 Methodology	1.4
1.5 Organization of the Report	1.5
1.6 Summary	1.6
1.7 Conclusions	1.7
1.8 References	1.8
1.9 Appendix	1.9
1.10 Bibliography	1.10
1.11 Glossary	1.11
1.12 Acronyms	1.12
1.13 Abbreviations	1.13
1.14 Symbols	1.14
1.15 Units	1.15
1.16 Notations	1.16
1.17 Figures	1.17
1.18 Tables	1.18
1.19 References	1.19
1.20 Appendix	1.20
1.21 Bibliography	1.21
1.22 Glossary	1.22
1.23 Acronyms	1.23
1.24 Abbreviations	1.24
1.25 Symbols	1.25
1.26 Units	1.26
1.27 Notations	1.27
1.28 Figures	1.28
1.29 Tables	1.29
1.30 References	1.30
1.31 Appendix	1.31
1.32 Bibliography	1.32
1.33 Glossary	1.33
1.34 Acronyms	1.34
1.35 Abbreviations	1.35
1.36 Symbols	1.36
1.37 Units	1.37
1.38 Notations	1.38
1.39 Figures	1.39
1.40 Tables	1.40
1.41 References	1.41
1.42 Appendix	1.42
1.43 Bibliography	1.43
1.44 Glossary	1.44
1.45 Acronyms	1.45
1.46 Abbreviations	1.46
1.47 Symbols	1.47
1.48 Units	1.48
1.49 Notations	1.49
1.50 Figures	1.50
1.51 Tables	1.51
1.52 References	1.52
1.53 Appendix	1.53
1.54 Bibliography	1.54
1.55 Glossary	1.55
1.56 Acronyms	1.56
1.57 Abbreviations	1.57
1.58 Symbols	1.58
1.59 Units	1.59
1.60 Notations	1.60
1.61 Figures	1.61
1.62 Tables	1.62
1.63 References	1.63
1.64 Appendix	1.64
1.65 Bibliography	1.65
1.66 Glossary	1.66
1.67 Acronyms	1.67
1.68 Abbreviations	1.68
1.69 Symbols	1.69
1.70 Units	1.70
1.71 Notations	1.71
1.72 Figures	1.72
1.73 Tables	1.73
1.74 References	1.74
1.75 Appendix	1.75
1.76 Bibliography	1.76
1.77 Glossary	1.77
1.78 Acronyms	1.78
1.79 Abbreviations	1.79
1.80 Symbols	1.80
1.81 Units	1.81
1.82 Notations	1.82
1.83 Figures	1.83
1.84 Tables	1.84
1.85 References	1.85
1.86 Appendix	1.86
1.87 Bibliography	1.87
1.88 Glossary	1.88
1.89 Acronyms	1.89
1.90 Abbreviations	1.90
1.91 Symbols	1.91
1.92 Units	1.92
1.93 Notations	1.93
1.94 Figures	1.94
1.95 Tables	1.95
1.96 References	1.96
1.97 Appendix	1.97
1.98 Bibliography	1.98
1.99 Glossary	1.99
1.100 Acronyms	1.100
1.101 Abbreviations	1.101
1.102 Symbols	1.102
1.103 Units	1.103
1.104 Notations	1.104
1.105 Figures	1.105
1.106 Tables	1.106
1.107 References	1.107
1.108 Appendix	1.108
1.109 Bibliography	1.109
1.110 Glossary	1.110
1.111 Acronyms	1.111
1.112 Abbreviations	1.112
1.113 Symbols	1.113
1.114 Units	1.114
1.115 Notations	1.115
1.116 Figures	1.116
1.117 Tables	1.117
1.118 References	1.118
1.119 Appendix	1.119
1.120 Bibliography	1.120
1.121 Glossary	1.121
1.122 Acronyms	1.122
1.123 Abbreviations	1.123
1.124 Symbols	1.124
1.125 Units	1.125
1.126 Notations	1.126
1.127 Figures	1.127
1.128 Tables	1.128
1.129 References	1.129
1.130 Appendix	1.130
1.131 Bibliography	1.131
1.132 Glossary	1.132
1.133 Acronyms	1.133
1.134 Abbreviations	1.134
1.135 Symbols	1.135
1.136 Units	1.136
1.137 Notations	1.137
1.138 Figures	1.138
1.139 Tables	1.139
1.140 References	1.140
1.141 Appendix	1.141
1.142 Bibliography	1.142
1.143 Glossary	1.143
1.144 Acronyms	1.144
1.145 Abbreviations	1.145
1.146 Symbols	1.146
1.147 Units	1.147
1.148 Notations	1.148
1.149 Figures	1.149
1.150 Tables	1.150
1.151 References	1.151
1.152 Appendix	1.152
1.153 Bibliography	1.153
1.154 Glossary	1.154
1.155 Acronyms	1.155
1.156 Abbreviations	1.156
1.157 Symbols	1.157
1.158 Units	1.158
1.159 Notations	1.159
1.160 Figures	1.160
1.161 Tables	1.161
1.162 References	1.162
1.163 Appendix	1.163
1.164 Bibliography	1.164
1.165 Glossary	1.165
1.166 Acronyms	1.166
1.167 Abbreviations	1.167
1.168 Symbols	1.168
1.169 Units	1.169
1.170 Notations	1.170
1.171 Figures	1.171
1.172 Tables	1.172
1.173 References	1.173
1.174 Appendix	1.174
1.175 Bibliography	1.175
1.176 Glossary	1.176
1.177 Acronyms	1.177
1.178 Abbreviations	1.178
1.179 Symbols	1.179
1.180 Units	1.180
1.181 Notations	1.181
1.182 Figures	1.182
1.183 Tables	1.183
1.184 References	1.184
1.185 Appendix	1.185
1.186 Bibliography	1.186
1.187 Glossary	1.187
1.188 Acronyms	1.188
1.189 Abbreviations	1.189
1.190 Symbols	1.190
1.191 Units	1.191
1.192 Notations	1.192
1.193 Figures	1.193
1.194 Tables	1.194
1.195 References	1.195
1.196 Appendix	1.196
1.197 Bibliography	1.197
1.198 Glossary	1.198
1.199 Acronyms	1.199
1.200 Abbreviations	1.200
1.201 Symbols	1.201
1.202 Units	1.202
1.203 Notations	1.203
1.204 Figures	1.204
1.205 Tables	1.205
1.206 References	1.206
1.207 Appendix	1.207
1.208 Bibliography	1.208
1.209 Glossary	1.209
1.210 Acronyms	1.210
1.211 Abbreviations	1.211
1.212 Symbols	1.212
1.213 Units	1.213
1.214 Notations	1.214
1.215 Figures	1.215
1.216 Tables	1.216
1.217 References	1.217
1.218 Appendix	1.218
1.219 Bibliography	1.219
1.220 Glossary	1.220
1.221 Acronyms	1.221
1.222 Abbreviations	1.222
1.223 Symbols	1.223
1.224 Units	1.224
1.225 Notations	1.225
1.226 Figures	1.226
1.227 Tables	1.227
1.228 References	1.228
1.229 Appendix	1.229
1.230 Bibliography	1.230
1.231 Glossary	1.231
1.232 Acronyms	1.232
1.233 Abbreviations	1.233
1.234 Symbols	1.234
1.235 Units	1.235
1.236 Notations	1.236
1.237 Figures	1.237
1.238 Tables	1.238
1.239 References	1.239
1.240 Appendix	1.240
1.241 Bibliography	1.241
1.242 Glossary	1.242
1.243 Acronyms	1.243
1.244 Abbreviations	1.244
1.245 Symbols	1.245
1.246 Units	1.246
1.247 Notations	1.247
1.248 Figures	1.248
1.249 Tables	1.249
1.250 References	1.250
1.251 Appendix	1.251
1.252 Bibliography	1.252
1.253 Glossary	1.253
1.254 Acronyms	1.254
1.255 Abbreviations	1.255
1.256 Symbols	1.256
1.257 Units	1.257
1.258 Notations	1.258
1.259 Figures	1.259
1.260 Tables	1.260
1.261 References	1.261
1.262 Appendix	1.262
1.263 Bibliography	1.263
1.264 Glossary	1.264
1.265 Acronyms	1.265
1.266 Abbreviations	1.266
1.267 Symbols	1.267
1.268 Units	1.268
1.269 Notations	1.269
1.270 Figures	1.270
1.271 Tables	1.271
1.272 References	1.272
1.273 Appendix	1.273
1.274 Bibliography	1.274
1.275 Glossary	1.275
1.276 Acronyms	1.276
1.277 Abbreviations	1.277
1.278 Symbols	1.278
1.279 Units	1.279
1.280 Notations	1.280
1.281 Figures	1.281
1.282 Tables	1.282
1.283 References	1.283
1.284 Appendix	1.284
1.285 Bibliography	1.285
1.286 Glossary	1.286
1.287 Acronyms	1.287
1.288 Abbreviations	1.288
1.289 Symbols	1.289
1.290 Units	1.290
1.291 Notations	1.291
1.292 Figures	1.292
1.293 Tables	1.293
1.294 References	1.294
1.295 Appendix	1.295
1.296 Bibliography	1.296
1.297 Glossary	1.297
1.298 Acronyms	1.298
1.299 Abbreviations	1.299
1.300 Symbols	1.299

LIST OF ACRONYMS AND ABBREVIATIONS

=====

=====

ML - Machine Learning

AI - Artificial Intelligence

DDoS - Distributed Denial Of Service

Fig. - Figure

Colab - Collaborator

IOT - Internet of things

OS - Operating System

KNN - K nearest neighbor

LIST OF FIGURES

Figure

Page No

3

4₂

2

5

2

6

14

15

7

11

9

21

10

22

17

22

12

23

13

23

14

24

24

16

25

17

26

26

19

27

20

27

21

28

21₅

29

23

29

18

24	30
25	31
26	31
27	33
	33
29	34
30	34
31	35
32	35
33	36
34	36
35	36
36	37
37	37

LIST OF GRAPHS

C'.raph

Page No

30

32

3

38

4

39

1

LIST OF TABLE

2

Table

Page No

16

2

20

ABSTRACT

I





—



Can I Recover My Files?

How Do I Pay?

Fig.

2

The cases of cyberattacks increased even more during the pandemic. Following image gives us a clearer picture about the same.



Fig. 3

Current static and dynamic analysis methods fail to detect malwares efficiently, particularly in case of zero-day attacks or polymorphic malwares. This led us to work towards utilizing

2



Malware are of many different types and each have their own specific purpose and functionalities. Here we discuss some of them -

Virus

They are the simplest malware and require human interaction to self replicate. These are capable of damaging the hardware as well as operating system. They are capable of performing following functions -

- Changing the file's size and/or deleting the files
- 0 Formatting the hard disk
- 0 Running down background operations and eventually slowing the system down

Computer worms are similar to viruses but the only difference is that they are able to execute on their own without any human interaction. They are able to spread via networks to other computers and replicate themselves in those machines. Worms are of many different types like -

- 0 E-mail Worms
- 0 File Worms shared in the network
- 0 Internet Worms

Rootkit

Rootkit enables attackers to do privilege escalation and gain administrative access, Hence the name. These are extremely difficult to detect and trace as -

- They can modify and alter the active processes to evade detection by not displaying the rootkit processes.
- 0 They are also able to hide themselves from antivirus by changing their signatures continuously.

Ransom ware

Ransomware prevents the user from accessing the computer by encrypting all the data present and the decryption is done only after a ransom is paid to the attacker through some cryptocurrency to avoid any detection from law enforcement agencies. They are able to -

- 0 Format the whole system in case the ransom is not paid
- 0 Stop all the anti-virus, anti-spyware or other security measures causing hindrance in its process

Spyware

As the name suggests, this malware is used to perform espionage i.e spy on the digital activities of the victim. It monitors and gathers information such as personal records, websites visited and the actions taken by them when they visit, their credentials etc via a hidden channel back to the attacker giving them a leverage over the victim. Sometimes these are also used by the organizations in order to target potential consumers for promotional purposes leading to increased number of spams. It is capable of following functions -

- It ends up showing unwanted ads and redirects the victim to desired third party websites.
- 0 It can cause undesirable changes in the system resulting in reduced security.
- 0 It can also act as a gateway to a backdoor giving hackers remote access to the system.

Adware

It can also be considered as a subclass of spyware and the only purpose of it is to display ads and increase traffic to a particular third party site. It also ends up using RAM and additional memory slowing down the system.

Trojan
This malware gets its name from a wooden horse used by Greeks in the Trojan war as just like that wooden horse, a seemingly harmless software or file contains a malicious hidden file and once executed, it enables the attacker to gain unauthorized access to the victim's computer.

Backdoor

As the name suggests, backdoor provides a secret hidden path to the attacker for gaining unauthorized access to the victim's computer. By itself it is completely harmless but provides a medium for various exploits and can also act as a zombie host used for DDoS attacks.

Keylogger

It logs all the keystrokes made by the user resulting in all the confidential data being compromised resulting in possible cases of blackmail or extortion. One of the advisable methods to protect ourselves from this is by using an onscreen keyboard while entering our credentials.

Remote Access Tool

It helps the attacker to escalate the privileges and provide remote access of the system to the attacker. Following functions can be performed -







for well known malware but they fail in case of polymorphic malware as they continuously keep on changing their signature.

Behaviour Detection

Also known as heuristics based analysis, it involves observing the behaviour of a file during execution and flagging it as malicious if found suspicious. Modifying of host files or registry keys may seem harmless but a combination of such activities is definitely a point of concern and any file exceeding a certain threshold raises an alert. The best way to implement this is via virtual environment. Although it will take more time, it is a much safer and foolproof option as compared to signature based detection.

Feature Detection

It can be seen as an application of derivative based detection and is able to overcome the false alarms associated with behaviour detection. This method is similar to identifying the anomalies but different also as it utilizes the characteristics which have been manually developed instead of using ml algorithms.

Machine Learning

The central idea of machine learning is to build a model that is capable of receiving input data and use its statistical analysis to predict the correct outcome. The model is capable of learning on its own. It can also be seen as a subclass of ai. The basic idea behind this is to train a model to produce some output based on a certain algorithm. A training dataset is provided and the model built on that dataset is used to make predictions.













These are the values which are wrongly predicted but is true in reality i.e. - when we have positive values of actual class but negation in predicted class.

- **False Negative (FN)**

These are the values which are wrongly predicted and negative in actual class.

selected elements

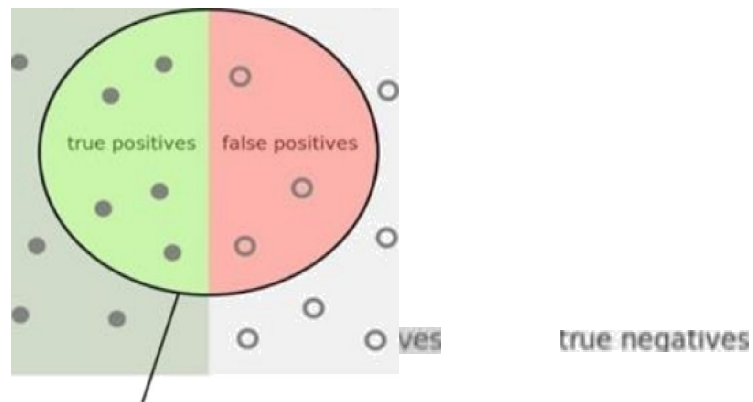


Fig. h

1.6 Organization

While going through this project you will mainly come across these components -

- **Chapter 1** gives us a basic idea about the project and helps you familiarize with the necessary technical and theoretical aspects of our project
- **Chapter 2** consists of review of other journals and research papers

G Chapter 3 tells us about the system design and various tools and techniques needed to achieve the same

- Chapter 4 gives us a comparative study of the performance of each model making it easier for us to choose the best one most suitable for us

G Chapter 5 provides us with the conclusion and also tells us about the future scope of our project

CHAPTER 2

LITERATURE SURVEY



Performance Performance		Classifier	KNN	SVM	Naive Bayes	J48	Random Forest
		Classifier					
CAS	Accuracy		87%	87.6%	72.34%	93.3%	95.69%
	Accuracy		94.6%	94.6%	55%	94.6%	96.8%
=	False-positives		12	20	0	15	9
	False-negatives		8	0	167	5	3
	True-positives		302	310	143	305	307
	True-negatives		49	41	61	46	52

Table 2

CHAPTER 3
SYSTEM DEVELOPMENT

The project was completed in the required time duration in a stepwise manner. Following timeline gives us a better understanding of the stages in which our project was divided and the time taken to complete that stage of development.

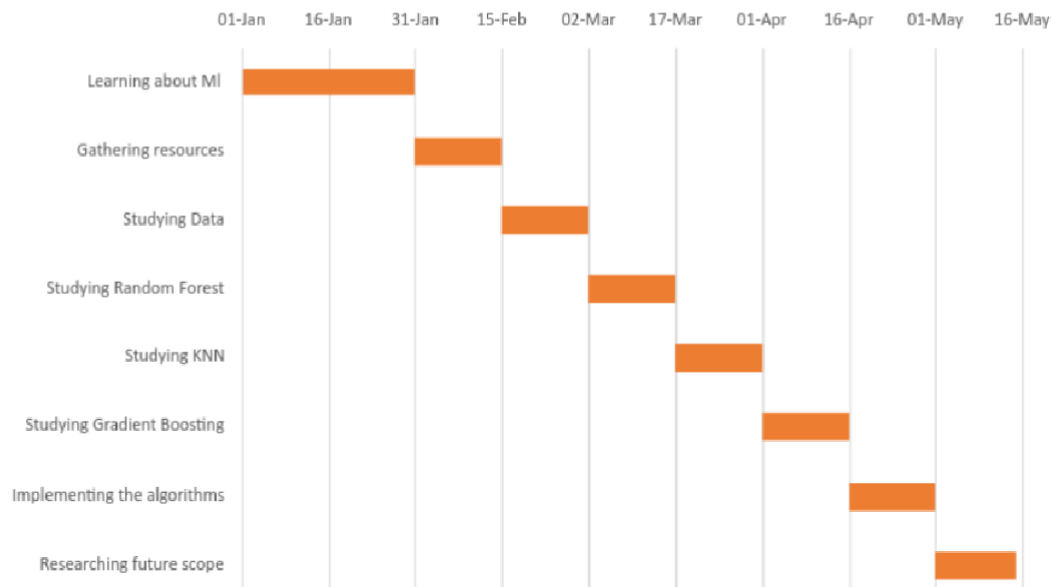


Fig. 9

We have used many Tools and technologies used for our project and all are discussed here -

3.1 Python

- Python is a high level, general purpose programming language developed in 1991 by Guido van Rossum.
- It is extremely favourable because of its code readability. It is an object oriented language having a large number of libraries and modules making it one of the go to languages in various fields.
- It has important libraries like Pandas and Numpy which are extensively used in the field of deep learning, machine learning and ai as they help in visualizing the data.



Fig. 1()

3d Google Colab

Google Colab is a powerful software made by Google Inc which is used by many Data Scientists for visualization of Data as well as preprocessing the data.

- It provides a free software platform where various Machine Learning algorithms can be implemented.

Free Access to Industry Grade GPU's and Cpu processing is provided by Google .

Fig. 11

3.3 Pandas

This happens to be the most sought after tool for data scientist

This python library is useful in data analysis and manipulation

It helps in manipulation, transformation, cleaning and analysis of data

Fig. 12

3.4 Numpy

- It is python library useful for numerical analysis
- The library consists of matrix ds and multidimensional arrays

Also has functions capable of performing linear algebra, fourier transformation and matrices

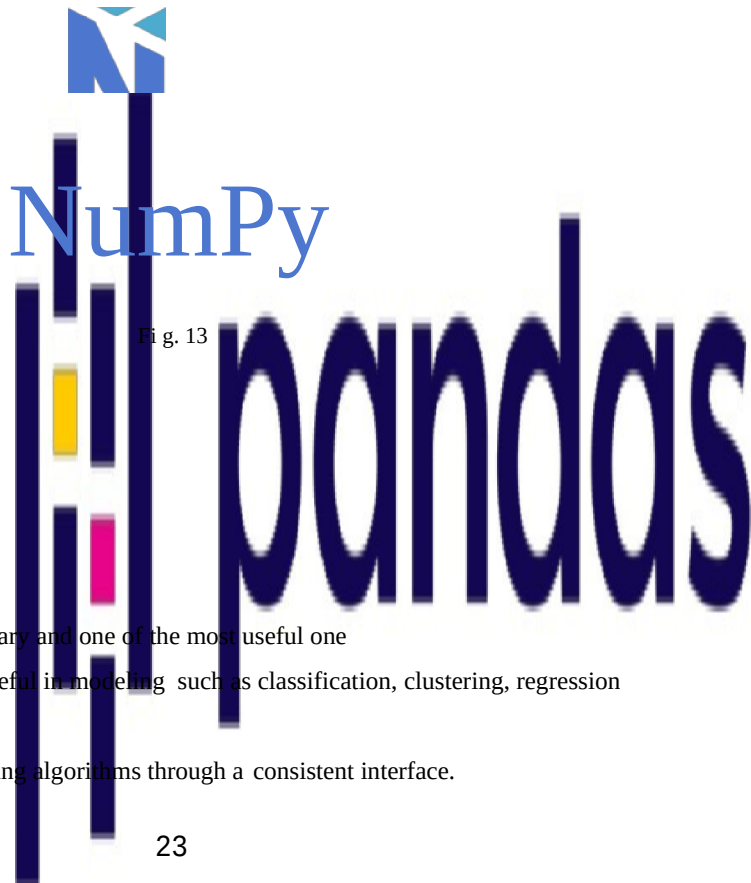


Fig. 13

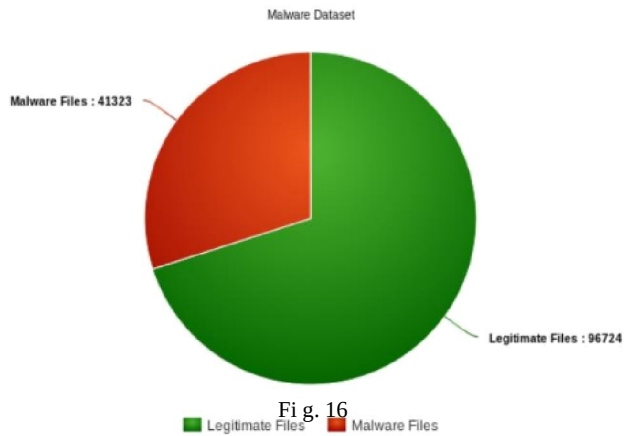
3.5 Scikit - learn

It is a free ml python library and one of the most useful one

It contains many tools useful in modeling such as classification, clustering, regression etc.

- It provides many learning algorithms through a consistent interface.





Fi g. 16

3.8 Implementation

As per our ongoing discussion, some of the security threats and challenges faced by the digital world have been discussed in depth. Keeping them in mind we will go with a static analysis method for achieving the desired outcome.

3.8.1 Preparing the dataset

- **Importing Dataset**

Since we are working on colab, the first steps towards processing our data and implementing the model would be to import the dataset.

```
from google.colab import files

uploaded = files.upload()

Choose files MalData.csv
• MalData.csv(text/csv) - 50668563 bytes, last modified: 12/05/2021 - 100% done
Saving MalData.csv to MalData.csv
```

Fig. 17

- Extracting t'eatures

Our dataset in total consists of 56 features.

```
Data columns (total 57 columns):
# Column
---
0 Name
1 md5
2 Machine
3 SizeOfOptionalHeader
4 Characteristics
5 MajorLinkerVersion
6 MinorLinkerVersion
7 SizeOfCode
8 SizeOfInitializedData
9 SizeOfUninitializedData
10 AddressOfEntryPoint
11 BaseOfCode
12 BaseOfData
13 ImageBase
14 SectionAlignment
15 FileAlignment
16 MajorOperatingSystemVersion
17 MinorOperatingSystemVersion
18 MajorImageVersion
19 MinorImageVersion
20 MajorSubsystemVersion
21 MinorSubsystemVersion
22 SizeOfImage
23 SizeOfHeaders
24 CheckSum
25 Subsystem
26 DLLCharacteristics
27 SizeOfStackReserve
28 SizeOfStackCommit
29 SizeOfHeapReserve
30 SizeOfHeapCommit
31 LoaderFlags
32 NumberOfRvaAndSizes
33 SectionsNb
```

Fig. 18

```

34 SectionsMeanEntropy
35 SectionsMinEntropy
36 SectionsMaxEntropy
37 SectionsMeanRawSize
38 SectionsMinRawSize
39 SectionMaxRawSize
40 SectionsMeanVirtualSize
41 SectionsMinVirtualSize
42 SectionMaxVirtualSize
43 ImportsNbDLL
44 ImportsNb
45 ImportsNbOrdinal
46 ExportNb
47 ResourcesNb
48 ResourcesMeanEntropy
49 ResourcesMinEntropy
50 ResourcesMaxEntropy
51 ResourcesMeanSize
52 ResourcesMinSize
53 ResourcesMaxSize
54 LoadConfigurationSize
55 VersionInformationSize
56 Legitimate

```

Fig. 19

As mentioned above our total dataset consists of 1,35,048 lines and the first 41,323 lines consists of legitimate files and their features whereas the rest of the dataset contains malicious files. Legitimate files have legitimate value as '1' whereas malicious files have legitimate value '0'.

```

Name md5 Machine SizeOfOptionalHeader \
2 setup.exe 4d92f518527353c8db88a78fdcdcf398 332 224
Characteristics MajorLinkerVersion MinorLinkerVersion SizeOfCode \
2 3330 9 0 517120
SizeOfInitializedData SizeOfUninitializedData AddressOfEntryPoint \
2 621568 0 350896
BaseOfCode BaseOfData ImageBase SectionAlignment FileAlignment \
2 4096 811808 771751936.0 4096 512
MajorOperatingSystemVersion MinorOperatingSystemVersion \
2 5 1
MajorImageVersion MinorImageVersion MajorSubsystemVersion \
2 0 0 5
MinorSubsystemVersion SizeOfImage SizeOfHeaders CheckSum Subsystem \
2 1 1150976 1024 1159017 2
DllCharacteristics SizeOfStackReserve SizeOfStackCommit \
2 32832 1048576 4096
SizeOfHeapReserve SizeOfHeapCommit LoaderFlags NumberOfRvaAndSizes \
2 1048576 4096 0 16
SectionsNb SectionsMeanEntropy SectionsMinEntropy SectionsMaxEntropy \
2 4 6.409558 4.805191 7.609957
SectionsMeanRawSize SectionsMinRawSize SectionMaxRawSize \
2 773488.0 21584 517120
SectionsMeanVirtualSize SectionsMinVirtualSize SectionMaxVirtualSize \
2 284498.0 21456 516760
ImportsNbDLL ImportsNb ImportsNbOrdinal ExportNb ResourcesNb \
2 14 235 21 1 11
ResourcesMeanEntropy ResourcesMinEntropy ResourcesMaxEntropy \
2 4.426324 2.846449 5.271013
ResourcesMeanSize ResourcesMinSize ResourcesMaxSize \
2 31102.272727 104 270376
LoadConfigurationSize VersionInformationSize legitimate
2 72 18 1

```

Fig. 20

In the above figure we can clearly see that the legitimate value is 1 which implies that the data is of a legitimate file.

● Organizing dataset

After importing the data and extracting the features, we organize the dataset into legit and mal sets.

```
maldata = pd.read_csv("MalData.csv", sep="|")
legit = maldata[0:41323].drop(["legitimate"], axis=1)
mal = maldata[41323:].drop(["legitimate"], axis=1)
print("The legit dataset is %s samples and %s features"%(legit.shape[0],legit.shape[1]))
print("The malware dataset is %s samples and %s features"%(mal.shape[0],mal.shape[1]))
The legit dataset is 41323 samples and 56 features
The malware dataset is 96724 samples and 56 features
```

Fig. 21

● Feature Selection

After organizing and dividing the dataset, we move towards selecting the most important features from our dataset. The dataset consists of 56 features but not all will be of that much importance. So we use tree based feature selection to assign weight to features and select the most important ones out from the 56 features.

```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
```

Fig. 22

We import the necessary libraries and proceed towards selecting the features.

```
data_in = maldata.drop(['Name', 'md5', 'legitimate'], axis =1).values
labels = maldata['legitimate'].values
feslec = ExtraTreesClassifier().fit(data_in, labels)
select = SelectFromModel(feslec, prefit = True)
data_in_new = select.transform(data_in)

print(data_in.shape, data_in_new.shape)
(138047, 54) (138047, 14)
```

Fig. 23

From the above output image we can see that out of the total 54 features (removing the name, md5 and legitimate column as they are not necessary in our scenario) only 14 were important and selected using the tree based feature selection. We can also see the features selected and the weight assigned to each one of them.

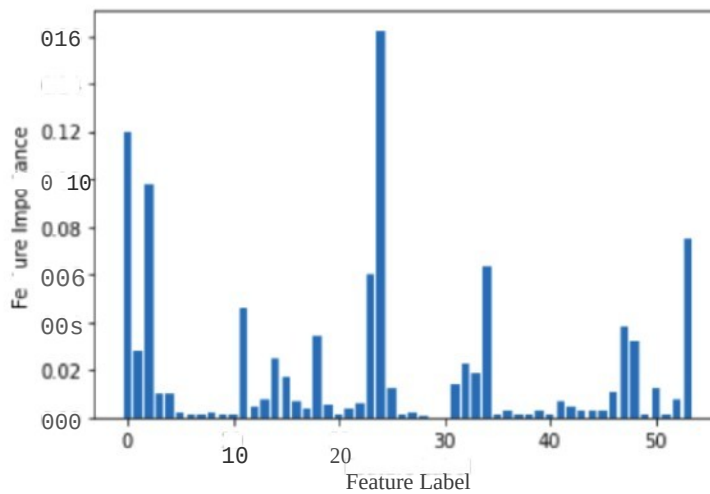
```

1 DllCharacteristics 0.16226741327135288
2 Machine 0.11935542710537982
3 Characteristics 0.0978731895331685
4 VersionInformationSize 0.07488837624889287
5 SectionsMaxEntropy 0.06293680781490484
6 Subsystem 0.059796529154792596
7 ImageBase 0.04579979836337387
8 ResourcesMinEntropy 0.03832223451688429
9 MajorSubsystemVersion 0.03435805162432689
10 ResourcesMaxEntropy 0.03214574948962828
11 SizeOfOptionalHeader 0.027777444767734083
12 MajorOperatingSystemVersion 0.024546137842613103
13 SectionsMeanEntropy 0.02255286537499676
14 SectionsMinEntropy 0.018657298140640655

```

Fig. 24

We get the list of features selected along with their weightage. 'DllCharacteristics' having the highest weightage of 0.16226741327135288 to 'SectionsMinEntropy' having the lowest weightage out of all the selected features i.e 0.018657298140640655. We can visualize this as a graph also.



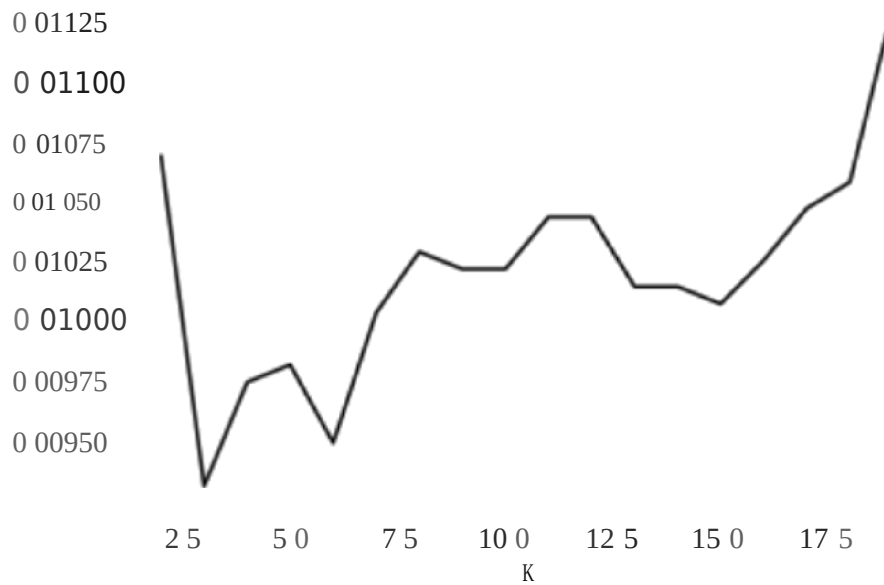
Graph 1

0 Splitting the dataset



- KNN

The next algorithm we are going to implement is knn. In order to implement this algorithm, we first need to identify the value of k or the nearest neighbours that will be considered for identifying the file. In order to obtain the most optimum value of k, we use the elbow curve method.



We check the error percentage for each value of k ranging from 1 to 20. See and the minimum error value is at k = 3. So we implement the knn algorithm using the value 3.

0.00925

```
knn = KNeighborsClassifier(n_neighbors=3, p=2)
knn.fit(legit_train,mal_train)

k =knn.score(legit_test,mal_test)*100
```

Fig. 27

- **Gradient Boosting**

In order to implement the gradient boosting algorithm, we need to specify the estimator. So we select the estimator value as 50 and implement the gradient boosting algorithm.

```
from sklearn.ensemble import GradientBoostingClassifier
grad_boost = GradientBoostingClassifier(n_estimators =50)
grad_boost.fit(legit_train,mal_train)
gb=grad_boost.score(legit_test,mal_test)*100
```

Fig. 28

CHAPTER 4

PERFORMANCE ANALYSIS

In order to compare and analyze the performance of our algorithms we take the use of confusion matrix and also take into consideration the accuracy of each algorithm.

4.1 Random Forest

Random forest was successfully implemented, so we look at the confusion matrix of the same.

```
from sklearn.metrics import confusion_matrix
result = classifier.predict(test_data)
conf_mat = confusion_matrix(true_labels, result)

print(conf_mat)

[[19305  84]
 [ 52 8169]]
```

Fig. 29

We get the above confusion matrix for random forest algorithm and as is visible from it, the false positives and false negatives are very low or negligible as compared to true positives or true negatives.

```
print("False Positives: ", conf_mat[0][1]/sum(conf_mat[0])*100)
print("False Negatives: ", conf_mat[1][0]/sum(conf_mat[1])*100)

False Positives: 0.4332353396255609
False Negatives: 0.6325264566354458
```

Fig. 30





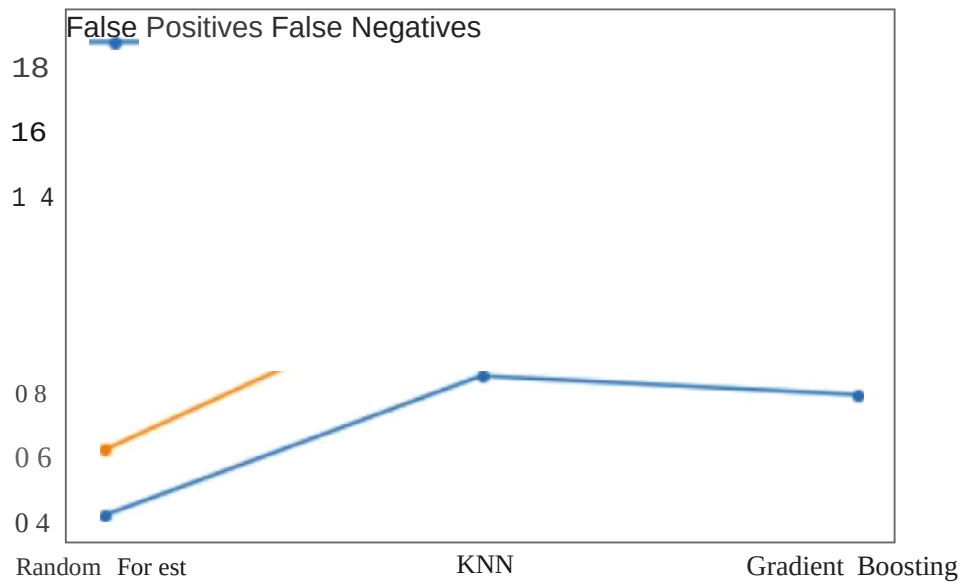


CHAPTER 5

CONCLUSIONS

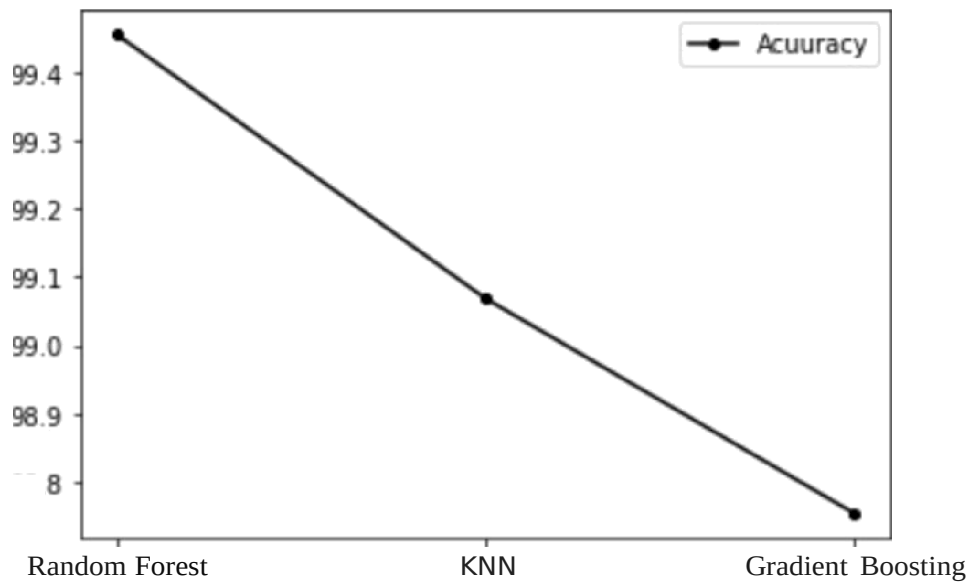
5.1 Conclusions

We plot a graph of false positives and false negatives for better visualization of the results.



Graph 3

Based on the graph we can conclude that random forest will have higher accuracy as compared to the other two algorithms. After successfully implementing all the algorithms we can infer that the highest accuracy of 99.45% was achieved by random forest followed by knn achieving an accuracy of 99.06%. Gradient boosting comes at last with an accuracy of 98.75%. The result can be conceptualized better as a graph.



Graph 4

Hence the best algorithm out of all the algorithms implemented for our project is random forest with an accuracy of 99.45%.

5d Future Scope

Considering the fast changing pace of technology a lot can be done to further improve upon our proposed model

The Algorithms implemented are sufficient by themselves but nowadays Neural Networks play an important role in classification problems.

- Neural Networks can be implemented instead of implementing machine learning algorithms which are much better in terms of unsupervised learning.
- Specific feature selection can also be done in order to remove false positives

5d Applications

G In the field of security this has wide and far reaching implications as it can be used to detect malicious files and many ay companies are working towards incorporating it in their software.

REFERENCES

1. Firdausi, I., Erwin, A. and Nugroho, A.S., 2010, December. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)* (pp. 201-203). **IEEE**.
2. Gavriluț, D., Cimpoșu, M., Anton, D. and Ciortuz, L., 2009, October. Malware detection using machine learning. In *2009 International Multi-Conference on Computer Science and Information Technology (IMCICT)* (pp. 735-741). **IEEE**.
3. Amos, B., Turner, H. and White, J., 2013, July. Applying machine learning classifiers to dynamic android malware detection at scale. In *2013 9th International Wireless Communications and Mobile Computing (IWCMC)* (pp. 1666-1671). **IEEE**.
4. Narudin, F.A., Feizollah, A., Anuar, N.B. and Gani, A., 2016. Evaluation of machine learning classifiers for mobile malware detection. *Journal of Computer Science*, 20(1), pp.343-357.
5. Ahmed, F., Hameed, H., Shafiq, M.J. and Farooq, M., 2009, November. Using spatio-temporal information in API calls with machine learning algorithms for malware detection. In *Proceedings of the 2nd ACM Workshop on Security in Artificial Intelligence* (pp. 55-62).
6. Santos, I., Devesa, J., Brezo, F., Nieves, J. and Bringas, P.G., 2013. Opern: A static-dynamic approach for machine-learning-based malware detection. In *International Conference on Intelligent and Trusted Systems (ITS)* (pp. 271-280). Springer, Berlin, Heidelberg.
7. Ham, H.S. and Choi, M.H., 2013, October. Analysis of malware detection performance using machine learning classifiers. In *2013 International Conference on Information Technology (ICIT)* (pp. 400-495). **IEEE**.

8 Reddy, Krishna Sandeep, and Arun Pujari. 2006. N-gram analysis for computer virus detection.

9 Hung, Pham Van. 2011. An approach to fast malware classification with machine learning techniques.

Dewesha_Malware_Detection.docx

ORIGINALITY REPORT

15%
SIMILARITY
INDEX

8%
INTERNET
SOURCES

6%
PUBLICATION
S

11%
STUDENT PAPERS

PRIMARY SOURCES

1 Submitted to Jaypee University of Information Technology **9%**
Student Paper

2 www.theseus.fi **2%**
Internet Source

3 "Information and Communications Security", Springer Science and Business Media LLC, 2020 **<1%**
Publication

4 docplayer.net **<1%**
Internet Source

5 Submitted to University of Bradford **<1%**
Student Paper

6 Submitted to Newman College **<1%**
Student Paper

7 Submitted to The New Art College **<1%**
Student Paper

8 Submitted to Coventry University **<1%**
Student Paper

<1%

%

9 Vaibhav Verdhan. "Supervised Learning with Python", Springer Science and Business Media LLC, 2020
Publication < 1
%

10 pt.scribd.com
Internet Source < 1
%

11 link.springer.com
Internet Source < 1
%

12 www.ijcrd.com
Internet Source < 1
%

13 www.scititles.com
Internet Source < 1
%

< 1
%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16 JUNE 2021.....

Type of Document (Tick): B.Tech Project

Name: Dewesha Sharma **Department:** Computer Science Engineering and Information Technology **Enrolment No** 171361 **Contact No.** 9816280648

E-mail. 171361@juitsolan.in **Name of the Supervisor:** Dr. Himanshu Jindal

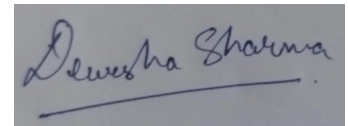
Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): MALWARE DETECTION USING MACHINE LEARNING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

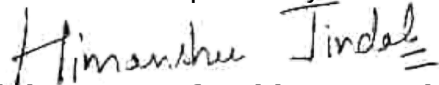
- Total No. of Pages = 53
- Total No. of Preliminary pages = 11
- Total No. of pages accommodate bibliography/references = 2



(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.



(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> ● All Preliminary Pages ● Bibliography/Images/Quotes ● 14 Words String 	15	Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through

the supervisor at plagcheck.juit@gmail.com