MALWARE DETECTION SYSTEM 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

Rajat Kumar Puri, 161269

Under the supervision of

Mr. Praveen Modi Assistant Professor



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

We hereby declare that the work presented in this report entitled "MALWARE **DETECTION SYSTEM USING MACHINE LEARNING**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is authentic record of our own work carried out over a period from August 2019 to December 2019 under the supervision of Mr.Praveen Modi(Assistant Professor), Computer Science and Engineering and information technology.

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

Rajat Kumar Puri,161269

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

dan 05-08-2020

Mr. Praveen Modi Assistant Professor Computer Science and Engineering/ Information Technology Dated:

Acknowledgement

We would like to express our special thanks of gratitude to our teacher and mentor **Mr**. **Praveen Modi** who gave us the great opportunity to do the project on the topic **MALWARE DETECTION SYSTEMUSING MACHINELEARNING**, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to him. Secondly, we would also like to thank Lab assistants who helped us a lot in finalizing this project within the limited time frame.

Rajat Kumar Puri, 161269

Table of Content

	ABSTRACT	9
1)	Chapter 1 –INTRODUCTION	10
	1.1 Introduction to the project	11
	1.2 Problem Statement	17
	1.3 Aims and Objectives	18
	1.4 Methodology	19
	1.5 Organization of Report	20
2)	Chapter 2 - LITERATURE REVIEW	21
	2.1 Malware analysis and classification	21
	2.2 Behavior based malware detection	23
3)	Chapter 3 - SYSTEM DESIGN	26
	3.1 System Requirements	26
	3.2 Why Python?	26
	3.3 Why ANACONDA?	26
	3.4 SCIKIT learn	27
	3.5 Pandas	27
4)	Chapter 4 – ALGORITHMS	28
5)	Chapter 5 – TEST PLAN	34
6)	Chapter 6 – RESULT AND ANALYSIS	38
7)	Chapter 7 - SYSTEM DESIGN	49
8)	REFRENCES	50

List of Abbreviation

- ML- Machine learning
- LR- Logistic Regression
- DL- Deep Learning
- KNN- K nearest neighbors
- SVM- Support Vector Machines
- AI- Artificial Intelligence
- CSV- Comma Separated Values
- CNN- Convolutional neural network
- OS Operating Systems

List of Figures

Page No

Fig 1.1	Introduction to ML	11
Fig 1.2	ML algorithm workflow	12
Fig 1.3	Types of machine learning.	13
Fig 1.4	Confusion Matrix	14
Fig 1.5	Pictorial representation of confusion matrix	15
Fig 1.6	Graphical representation of total malware infection growth rate	17
Fig 2.1	Overview of the research methodology.	23
Fig 2.2	Classifier performance comparison without feature selection	24
Fig 2.3	Classifier performance comparison with feature selection	24
Fig 4.1	Diagram of decision tree	29
Fig 4.2	Diagram of random forest	30
Fig 4.3:	Pictorial representation of SVM.	31
Fig 4.4	Pictorial representation of Logistic Regression	33
Fig 5.1:	Created dataset	35
Fig 5.2:	Created dataset with feature scaling	36
Fig 5.3:	Result without feature scaling	36
Fig 5.4:	Result with feature scaling	37
Fig 6.1:	Decision tree confusion matrix	40
Fig 6.2	Random Forest confusion matrix	40
Fig 6.3	XGBoost confusion matrix	41
Fig 6.4	KNN confusion matrix	41
Fig 6.5	Logistic Regression confusion matrix	42
Fig 6.6	SVM confusion matrix	42
Fig 6.7	Gaussian Naïve Bayes confusion matrix	43
Fig 6.8	Feature extracted Decision Tree confusion matrix	44
Fig 6.9	Feature extracted Random Forest confusion matrix	44

Figure No.

Description

Fig 6.10	Feature extracted XGBoost confusion matrix	45
Fig 6.11	Feature extracted KNN confusion matrix	45
Fig 6.12	Feature extracted Logistic Regression confusion matrix	46
Fig 6.13	Feature extracted SVM confusion matrix	46
Fig 6.14	Feature extracted Naïve Bayes confusion matrix	47

List of Tables

Table No	Description	Page No.
Table 4.1:	Difference between various ML Algorithms	33
Table 6.1:	Results without feature extraction	38
Table 6.2	Results with feature extraction	39

Abstract

Malware is risky these day and age for those web clients. It can compromise the host pc. By polymorphic malware we mean the one that changes its signature regularly to fool detection. We can define malware as piece of code or malicious code that harm the data or device. here, we made a alternative of virus detection by using machine learning techniques and created a dataset and used machine learning algorithms for categorizing the file into malicious or not and compared their results to determine the best algorithm suiting for our dataset.

<u>Chapter-1</u> <u>INTRODUCTION</u>

As the internet users increasing day by day Malware become the major problem in the field of internet. We can define the malware as a malicious code that can harm the file or software in the computer.

As the technology will increasing day by day the types of malwares also increasing and become more powerful from the previous ones. According to a research around 3900 different malware objects were identified. Expertise plays a important role for handling the malware as the malware are polymorphic now a days so there is a need of expert algorithms that will detect the malware accurately.

Now days it is very difficult to handle the malware for most of the companies because the samples of the malware increased very rapidly. There are around four lakhs malware samples so protection from the malware become the major task because due to malware a big amount of data will lost that is beneficial for us.

There are many ways to deal with the malware. The techniques are static analyses, dynamic analyses, and at last machine learning for detecting malware. In our project we focus on the last techniques. we use different types of algorithms for malware detection such as KNN ,decision tree ,Logistic Regression etc. and analyze the result of all the algorithms at the last.

(1.1) Terminologies

1. Machine Learning- These algorithms are a type of algorithms that makes the system or the software application to be smart enough to be able to more accurate without being explicitly programmed and can predict outcomes. The main idea behind these types of algorithms is that it receives input data in the form of text or images and the system or the model is trained with the statistical inputs to identify or predict the output and even updated the outputs as new data becomes available. It requires the algorithm to search through the dataset and look for patterns or similarities and manipulating or adjusting the system accordingly.



Fig 1.1: Introduction to ML [1]

2. **How ML works -** The procedure of machine learning begins with the assortment of information or perceptions as the info dataset which can be as pictures, content, tables and so on. Further, numerous predefined AI calculations are applied to the info information which either characterize the information into gatherings or distinguishes designs among the dataset to anticipate the yield and give fitting outcomes.



Fig 1.2: ML algorithm workflow [1]

- 3. Types of machine learning
 - **Supervised learning-**here algorithms works for a dataset that is as of now being prepared by past yields and results of the past utilizing marked information to foresee the result of the new information. It can also analyze the data and the outcome and compare it with the previously stored data to find errors and to be able to modify and train the model accordingly.
 - Unsupervised learning- it is different from the supervised learning as it is used to calculate result where the target value is not provided and we have to make a prediction. The most form of unsupervised learning is cluster analysis which used to analyze hidden pattern and helps in maintaining the data into the groups.
 - Semi supervised learning-it is the mix of both learning's that discuss above for training the datasets and produce much productive and powerful clusters. the model uses both labeled and unlabelled data for the training and it mostly need a small quantity of labeled data and a relatively huge quantity of unlabelled data which are used simultaneously to train the model.
 - Reinforcement learning-basically it is a reward based learning in which the

model will interact with the environment by doing action and discovering error or rewards. We can say that in this model learns from its mistakes and maximize the performance.



Fig 1.3: Types of machine learning [1]

4. **Interpretation of Performance Measures**- We can evaluate the performance in many ways. We can find the performance with the help of confusion matrix.

	Predicted						
Actual		Positive	Negative				
, local	Positive	True Positive (TP)	False Negative (FN)				
	Negative	False Positive (FP)	True Negative (TN)				

Fig 1.4: Confusion Matrix [2]

Confusion matrix has four tables it is type of two way table as shown in the diagram. The 2 sections in the green are the true positive and crimson is true negative and the results which are correctly predicted. The other 2 sections are in crimson are wrong because these values are wrongly calculated and thus needs to be decreased. These 2 sections are called as false negative and false positive respectively and this happens when there is a contradiction between true class and the calculated class.

- **True Positive**–This is the value that is correctly calculated and is positive value which can be defined as the +ve value of true class and +ve value of calculated class. It is shown with TP.
- **True Negative** This is the value which is correctly calculated but negative result which refers to the negation of true class and negation of calculated class. It is shown by TN.
- False Positive–This is the value which is wrongly calculated but is true in existence that is when we have true values of actual class but negation in calculated class.
- False Negative This is the value which is calculated wrong and -ve in true class.



Fig 1.5: Pictorial representation of confusion matrix [3]

• Accuracy – Accuracy is the most common execution measurement and it is specifically a proportion of effectively calculated perception to the aggregate perceptions. Some might imagine that on the chances of high trueness, our model is good. Truly exactness a unique quantity yet just you have symmetrical data where estimations of FP and FN are relatively same. In this manner, you need to take a gander at different parameters to achieve the execution of your model. For our model, we have 0.982939 which equals to 98%.

Accuracy = TrueP + TrueN / TrueP + FalseP + FalseN + TrueN

• **Precision**-Precision is the ratio of accurately found positive perceptions to the aggregate calculated +ve perceptions .The inquiry that this acquired answer is of all travelers with name dasendure, what number of really endure? High precision shows with the low false positive rate.

Precision = TrueP/TrueP+FalseP

• **Recall (Sensitivity)** - The Recall actually calculates how much of the true Positives our model calculates through labeling it as true (True Positive). Applying the same knowledge, we know that the Recall should be the model metric we used to choose our best model when there is a high price associated with False Negative.

Recall = TrueP/TrueP+FalseN

• **F1Score**- It is the weighted normal of Precision with addition to Recall. Consequently, this score finds both FP and FN. Instinctively it isn't as right as exactness, yet F1 is normally more essential than precision, mainly on the off chance of odd class. Precision works good if FP, FN have comparable expense. On the off chance that the price of FP and FN are together dissimilar, it is good to take a gander at both Precision with Recall.

F1 Score = 2*(R * P) / (R + P)

(1.2) Problem Statement

With the development of innovation, the quantity of malware is likewise expanding step by step. Malware now are structured with transformation trademark which causes a huge development in number of the variety of malware (Ahmadi, M. et al., 2016). Not just that, with the assistance of robotized malware created apparatuses, beginner malware creator is currently ready to effortlessly produce another variety of malware (Lanzi, A. et al., 2010). With these developments in new malware, conventional mark based malware identification are demonstrated to be incapable against the huge variety of malware (Feng, Z. et al., 2015). Then again, AI strategies for malware recognition are demonstrated powerful against new malwares. Simultaneously, AI strategies for malware identification have a high false positive rate for identifying malware (Feng, Z. et al., 2015).So we have to achieve the false rate as low as possible with machine leaning Algorithms.



Total Malware Infection Growth Rate (In Millions)

Fig 1.6: Graphical representation of total malware infection growth rate [4]

(1.3)Aims and Objectives

The aim of the project is to the purpose of this project is to get the best accuracy for malware detection and also to get the best algorithm which provides the result on this dataset.

Objectives-

- Download the dataset which contain both malicious and benign file and list them into CSVfile.
- Use the various ML algorithms and compare the algorithms on the behalf of parameters like accuracy, precision and reduce the FP rate.
- Use various methods such as feature extraction, feature scaling, to improve the accuracy and to avoid over fitting.

(1.4)Methodology

Malware detection is the most crucial step in securing the host computers. There are many machine learning algorithms that are important in the machine learning and we classified them into many techniques some of the examples are SVM, K-nearest neighbors clustering, Decision Trees etc. Our main aim is to find out whether a file contains malware or not.



We gathered the dataset from kaggle. As our problem statement comes under the category of classification. It is a single class classification in which a file is

- 1. Malicious
- 2. Benign

So our motive is to classify the following dataset into these classifications. Firstly we analyze the dataset. As there are so many columns in the dataset we use different techniques such as feature extraction, feature scaling etc. to avoid over fitting and to improve the accuracy of the project.

We can train our model in different ways. It depends how the dataset appeared after applying feature extraction and feature scaling techniques. we used

- 1. Decision Tree
- 2. XG Boost
- 3. KNN
- 4. Random Forest
- 5. Logistic regression

(1.5) Organization of Report

In Chapter 2, literature review, we study about two research paper. First of all there is a abstract of research paper then display some figure related to this and at last we write conclusion in which we saw the accuracy of the algorithms.

In Chapter 3, in this we write the system requirement so we can run all the algorithms and here we discuss all the libraries needed and why these libraries needed.

In Chapter 4, it is the algorithm part here we list all the algorithms we used in the project, so we can apply these algorithms in the project and get better results.

In Chapter 5,here we discuss about the dataset and the observation towards our result such as accuracy in all the algorithms.

InChapter6,here we compare the result of each algorithms without feature extraction and then with feature extraction and see which algorithm is best suited that is having high accuracy.

InChapter7, At last we concluded all we did in the project and more over we discuss the future work also such as we run the malware on the virtual environment.

<u>Chapter-2</u> <u>LITERATUREREVIEW</u>

(2.1)EktaGandotra, DivyaBansal, SanjeevSofat "Malware analysis and classification: A survey, Journal of Information Security, 2014" [5]

The greatest danger on internet is the safety of the user pc they over and over again get attacked by a malicious code infected file. The malwares being planned by aggressors are polymorphic and changeable which can change their code as they engender. It has become very difficult to keep track of all these new malwares and provide the customer with protection against all these new types of attacks by traditional methods.

ML for recognizing Malwares – In this we use various types of algorithms such as SVM, Decision Tree, Random Forest, Naive Bayes and Clustering. The result that is obtained gave insight that the best machine learning algorithm is a J48 decision tree.

Dataset and Methodology-dataset is small about 220 malicious samples or 250 Benign samples with and without feature selection. They are using five different Classifiers. Names of the classifiers are SVM, KNN, J48 decision tree and SVM.

Result- the best machine learning algorithm is a J48 decision tree providing a recall of 96.01%, a FPR of 3.39%, a precision of 96.97%, and an accuracy of 97.2%.

Now a days with increasing population in the developed malwares it has possessed a great danger to various online resources like host connected to internet or the web servers, etc. Identifying malwares on the basis of their digital signature is not a very efficient approach we have used machine learning approach. Firstly we analyze the malware file size and signature and apply static approach and then we apply dynamic if the classification could not be performed. The techniques that are developed are not sufficient to rectify the problem of detecting and removing the malwares efficiently without much adverse effect and some better methods needs to be developed to solve this issue.

(2.2)Ivan Firdausi, Charles Lim "ANALYSIS OF VARIOUS MACHINE LEARNING TECHNIQUES USED FOR BEHAVIOR-BASED MALWARE DETECTION"[6]

The increasing amount of malware that are coming daily became a greatest computer threat. Manually correcting is no longer taken as effective and efficient or good as compared against the high increasing rate of malware. Therefore dynamic or automatic malware detection on the basis of behavior using ML techniques is taken to be the best in the market.



Fig 2.2 Overview of the research methodology [6]



Fig 2.3 classifier performance comparison without feature selection [6]



Fig 2.4 classifier performance comparison with feature selection [6]

we can be say that By adding feature selection, the features were decreased to a large amount such as attributes reduced from 5191 to 116 in binary weighted data set and the time taken for training and building the model got shorter at the expense of the performance depreciation slightly. In some cases, the performance of the project can also get hiked a little bit. The evaluation of five unique models was also presented. The overall best accuracy was achieved by the decision tree called J48 decision tree using frequency-weight and not using feature selection data set, with a TP rate of 94.6%, a FPR of 3.6%, a +ve predictive value of 96.76%, and a score of 98.02%. The examination of the tests and experimental reports concluded that this approach is very useful to find out malware.

<u>Chapter-3</u> SYSTEM DESIGN

(3.1) System Requirements

Algorithms being implemented in this project requires some generic system for the processing of algorithms.

- Windows 10 (64-bit)
- ANACONDA
- Python
- 8 GBRAM
- Intel(R) Core(TM) i5-6200U CPU @ 2.50GHz

(3.2) Why Python

Python is a programming language with a huge group of spectators and it is exceptionally straightforward and can be effectively coherent. Moreover, python offers the assortment of bundles which makes the most scary calculations or ventures more straightforward. Python has libraries for pretty much every usable record for example - with working with pictures, working with content or working with audio records. In any event, when working with another OS, python is truly pliable. Python has a huge network which makes it simpler to look for help and tips and tricks.

(3.3) Why ANACONDA

ANACONDA is widely popular as it provides all the libraries pre-installed and make the user free from hassle of the otherwise installing all libraries. Approximately it has 100 packages which can be used for data science, machine learning or statistical analysis.

(3.4) SCIKIT LEARN

it is in python usually used for machine learning and create a lot of features for ML algorithms and in statistical modeling like regression, classification, clustering.

(3.5) PANDAS

It is software library offer a data structures and perform various operations like tables time series. It provides high-performance data manipulation and analysis tool using its powerful data structures. It used in various domains like finance, Analytics, Statistics etc.

<u>Chapter-4</u> ALGORITHMS

This section contains of various machine learning algorithms which are to be used in the project and discussed-

Supervised learning

• **K-Nearest Neighbors-** This algorithm used for both for the regression tasks as well as classification tasks but for most of the time in classification problems. This algorithm quite easy in implementation and need very less computation time and therefore is widely used machine learning algorithm. K in this algorithm stands for the number of neighbors which are specified by the user. In this the mathematical formula used to measure the K-nearest neighbors of the data points and then depending on the classes to which these data points belong it makes the prediction of the output.

Distance Functions

Euclidean Manhattan Minkowski $\begin{aligned}
\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \\
\sum_{i=1}^{k} |x_i - y_i| \\
\left(\sum_{i=1}^{k} (|x_i - y_i|)^q\right)^{1/q}
\end{aligned}$ • **Decision Trees-** This it is the type of algorithms in which we use tree data structure for the solution of the given problem. In this algorithm the terminal node stands for the any given class or category and the internal nodes between the root and the leaf are the nodes which define the attributes. The prediction that is made is based on some series of questions based on the features and upon reaching the terminal node by following the questions from root the final leaf node is the required class.

Formulation-



Fig 4.1 diagram of decision tree[7]

• Random Forest- Random forest belongs to the ensemble category of machine learning. In ensemble method several machine learning models are used to make a decision. In this algorithm several decision trees are created to make the classification or regression tasks. In this several decision trees are made and each of them is slightly different from the other this uniqueness is achieved by two methods firstly by not selecting all the data points in decision making instead using only some data points and then repeating some of them to make the count equal and the other method is not selecting all the features instead selecting only a subset of features to make different predictions and then taking average of all the decision trees. In this way the problem of over fitting which is in decision trees is also overcome. Here nestimators shows the number of decision trees to be made and maxfeatures shows the number of features to be selected for the subset.



Fig4.2diagram of random forest

• **Support Vector Machines-** It is mostly applied for classification problems in this search data point is plotted in n dimensional space and n equals to number of features present and the value that each feature represents is the value of each coordinate. separate hyper plane is used to differ as a boundary for classes.we can also call it support vector network.it can solve both linear and non linear problems.



Fig 4.3: Pictorial representation of SVM [8]

• **XGBoost-** full form is extreme gradient boosting technique. It is ensemble machine learning technique which is an implementation of gradient boosted decision trees. In gradient boosted decision trees several decision trees are made and instead of randomly selecting the features the succeeding trees learn from the preceding trees to improve the accuracy of the algorithm, in order to achieve this the decision trees are mostly prepared shallow with depth of around 5 in order for easy interpretation it also has a learning rate which defines how much a tree will learn from its preceding tree. The height and learning rate parameters are mostly set low in order to achieve and to reduce over fitting to a large extent.

• Naïve Bayes- It is a classification technique based on Bayes theorem. It thinks that one feature is not in relation with other features. eg. We take an example of apple if it is red, round and has a diameter of around 3 inches. All these factoring individually contribute to the probability that it is an apple. Due to this property it is known as naïve. Bernoulli Naïve Bayes Algorithm is used to binary classification problems. Gaussian Naïve Bayes used for normal classification problem but it is popular.

Formula of Bayes theorem used in Naïve Bayes



Posterior Probability Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \ldots \times P(x_n|c) \times P(c)$$

• **Logistic Regression-** -.it is the simplest for applying to binary classification. in short LR take the features values and calculate the probability using sigmoid or softmax function.



Logistic Regression for Binary Classification

Fig 4.4: Pictorial representation of Logistic Regression

Algorithms	Problem Type	Average predictive	Training	Prediction speed
		accuracy	speed	
KNN	Either	Lower	Fast	Depend on (n)
Linear Regression	Regression	Lower	Fast	Fast
Logistic Regression	Classification	Lower	Fast	Fast
Naïve Bayes	Classification	Lower	Fast	Fast
Decision Trees	Either	Lower	Fast	Fast
Random Forest	Either	Higher	Slow	Moderate

Table 4.1: Difference between various ML Algorithms

<u>Chapter-5</u> <u>TEST PLAN</u>

Here, we focused on working out our algorithms on the project to apply the machine learning algorithms and to carry out comparative analysis and store the results for further analysis. we have a dataset with 10539 rows and 57 columns.

Features:

'Machine', 'SizeOfOptionalHeader', 'Characteristics', 'MajorLinkerVersion' , 'MinorLinkerVersion', 'SizeOfCode', 'SizeOfInitializedData', 'SizeOfUninitializedData''AddressOfEntryPoint', 'BaseOfCode', 'BaseOfData', 'ImageBase', 'SectionAlignment', 'FileAlignment', 'MajorOperatingSystemVersion', 'MinorOperatingSystemVersion', 'Major ImageVersion', 'MinorImageVersion', 'MajorSubsystemVersion', 'MinorS ubsystemVersion', 'SizeOfImage', 'SizeOfHeaders', 'CheckSum', 'Subsystem', 'DllCharacteristics', 'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve', 'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'SectionsNb', 'SectionsMeanEntropy', 'SectionsMinEntropy', 'SectionsMaxEntropy', 'SectionsMeanRawsize', 'SectionsMinRawsize', 'SectionMaxRawsize', 'SectionsMeanVirtualsize', 'SectionsMinVirtualsize', 'SectionMaxVirtualsize', 'ImportsNbDLL', 'ImportsNb', 'ImportsNbOrdinal', 'ExportNb', 'ResourcesNb', 'ResourcesMeanEntropy', ResourcesMinEntropy', 'ResourcesMaxEntropy', 'ResourcesMeanSize', ResourcesMinSize', 'ResourcesMaxSize', 'LoadConfigurationSize', 'VersionInformationSize'.

Activities						Fri Nov 29, 10:43 PM						
						dataset - D	DataFrame					
Index	Name	md5	Machine	eOfOptionalHea	Characteristics	ajorLinkerV	ersic inorLinkerVersic	SizeOfCode	eOfInitialize	dDa:OfUninitializ	edE ldressOfEntry	yPoi
0	Windows.Int	09e83f1d1c9	34404	240	8226	14	12	779776	253952	0	56256	4
1	hidserv.dll	3030f19c6a7	34404	240	8226	14	12	21504	13312	Θ	22816	4
2	DmApiSetExt	8271846f8f5	34404	240	8226	14	12	33792	27648	Θ	32192	4
3	FSResizerSe	5802b421556	332	224	271	6	0	23552	164864	1024	12515	4
4	asc-setup.e	8cb1fb45489	332	224	33167	2	25	87040	71680	Θ	91076	4
5	PeerDistHtt	ff42a597ecd	34404	240	8226	14	12	38912	16384	0	38768	4
6	shutdownux	d38dfef6c48	34404	240	8226	14	12	104448	176640	Θ	100768	4
7	OnlineArmor	d69d127fb52	332	224	33167	2	25	37888	17920	0	39960	4
8	tapiperf.dll	383af082659	34404	240	8226	14	12	5120	7680	0	6208	4
9	tscfgwmi.dll	5d6c8631b0b	34404	240	8226	14	12	130048	77312	Θ	127776	4
10	mcupdate_Au	365dd269e50	34404	240	34	14	12	4608	91136	Θ	110608	4
11	dafpos.dll	d65a5fd868d	34404	240	8226	14	12	202752	88064	Θ	189424	4
12	httpprxc.dll	400cb5e63b7	34404	240	8226	14	12	7168	11776	Θ	8528	4
13	Scrivener-0	d18b0589dc5	332	224	271	6	0	602112	49152	1445888	2047984	1
14	upnp.dll	3445b6e05d8	34404	240	8226	14	12	225280	165376	0	18544	4
15	mbussdapi.d	2d7ab2226e6	34404	240	8226	14	12	52736	29696	Θ	51792	4
16	Chandler_wi	c1cc014a9a8	332	224	271	6	0	23040	119808	1024	12491	4
17	comcat.dll	590c68e5aec	34404	240	8226	14	12	3584	6656	Θ	4768	4
18	Windows.UI	49a78f5dfeb	34404	240	8226	14	12	Θ	612864	Θ	0	4
19	appmgr.dll	236316a1fbc	34404	240	8226	14	12	223232	236032	0	204560	4
4												-1

Fig 5.1: Created dataset

First of all we remove first two columns as they have categorical values. Then we applied feature scaling using algorithms called standard scalar. By this the mean of all the columns became zero and standard deviation became one.

Activiti	Activities Fri Nov 29, 10:43 PM									•∳ ₿ -			
						X_train	- NumPy array						
	0	1	2	3	4	5	6	7	8	9	10	11	12 🔺
0	-0.665677	-0.654523	-0.618849	0.668023	-0.710192	-0.0349932	-0.0384227	-0.013322	-0.198884	-0.0545115	-0.0351425	-0.653887	-0.24438
1	1.50223	1.4657	0.164965	0.398797	0.286981	-0.0277247	-0.0341763	-0.0207522	-0.0400486	-0.0545115	-0.0351425	1.27654	-0.24438
2	-0.665677	-0.654523	-0.618849	0.398797	-0.800844	-0.0342783	0.0460876	-0.0207522	-0.198699	-0.0545115	-0.0327705	-0.653887	-0.24438
3	1.50223	1.4657	0.164965	0.398797	0.286981	-0.0302985	-0.0189962	-0.0207522	-0.178274	-0.0545115	-0.0351425	1.55808	-0.24438
4	1.50223	1.4657	0.164965	0.398797	0.286981	-0.0200631	-0.0247305	-0.0207522	0.119626	-0.0545115	-0.0351425	1.55808	-0.24438
5	1.50223	1.4657	0.164965	0.398797	0.286981	-0.0350408	-0.0381051	-0.0207522	-0.195559	-0.0545115	-0.0351425	1.55808	-0.24438
6	-0.665677	-0.654523	-0.568385	-1.08194	-0.800844	-0.00182042	-0.038289	0.107651	1.71774	2.04846	0.143749	-0.653887	-0.24438
7	-0.665677	-0.654523	-0.618849	0.398797	0.196329	-0.0286065	-0.00430093	-0.0207522	-0.123282	-0.0545115	-0.0211079	-0.653887	-0.24438
8	1.50223	1.4657	0.164965	0.398797	0.286981	-0.0275579	-0.0334741	-0.0207522	-0.0298134	-0.0545115	-0.0351425	1.55808	-0.24438
9	-0.665677	-0.654523	-0.61757	-0.678105	-0.800844	-0.0338493	-0.037219	-0.0207522	-0.181834	-0.0545115	-0.0319798	-0.653887	-0.24438
10	-0.665677	-0.654523	-0.618849	0.398797	-0.800844	-0.0333727	0.010712	-0.0207522	-0.182479	-0.0545115	-0.0309914	-0.653887	-0.24438
11	-0.665677	-0.654523	-0.618849	0.398797	0.196329	-0.0286065	-0.00430093	-0.0207522	-0.123282	-0.0545115	-0.0211079	-0.653887	-0.24438
12	-0.665677	-0.654523	2.61832	-1.21656	1.46546	-0.0249603	0.00872253	-0.0207522	0.0300904	-0.0545115	-0.0133988	-0.653887	-0.24438
13	-0.665677	-0.654523	-0.618849	-0.274267	-0.800844	-0.00551423	0.00185136	-0.0207522	0.369937	-0.0545115	0.026926	-0.653887	-0.24438
14	-0.665677	-0.654523	2.61842	-1.21656	1.46546	-0.0344093	-0.0378543	-0.0207522	-0.181131	-0.0545115	-0.0329681	-0.653887	-0.24438
15	-0.665677	-0.654523	2.61832	-1.21656	1.46546	-0.0329914	-0.0275893	-0.0207522	3.25308	-0.0545115	-0.0302007	-0.653887	-0.24438
16	-0.665677	-0.654523	-0.618849	0.398797	0.196329	-0.0291784	-0.0101189	-0.0207522	-0.132798	-0.0545115	-0.0222939	-0.653887	-0.24438
17	-0.665677	-0.654523	2.61832	-1.21656	1.46546	-0.0249603	0.00872253	-0.0207522	0.0300904	-0.0545115	-0.0133988	-0.653887	-0.24438
18	-0.665677	-0.654523	-0.61757	-0.678105	-0.800844	-0.034731	-0.0345107	-0.0206941	-0.194947	-0.0545115	-0.0337588	-0.653887	-0.24438-
4													•
For	nat Resiz	e 🗸 Backgro	ound color										
													e Close

Fig 5.2: Created dataset with feature scaling

After that we apply the entire algorithm without feature extraction such as logistic regression, decision tree, XGBoost, Random Forest, KNN .and then compare the result of all the algorithms. The 75% of the data used to train the model and remaining 25% data is used to test the model in all the algorithms.

Console 1/A ×							2	
Accuracy on de Accuracy on de Accuracy with Accuracy with Accuracy with Accuracy with Accuracy with Accuracy with C:\Users\acer\ ConvergenceWar STOP: TOTAL NO Increase the m https://sc Please also re https://sc Please also re https://sc extra_warnin Accuracy with Accuracy with Accuracy with Accuracy with Accuracy with Accuracy with	cision tree training set: cision tree test set: 0.9 random forest classifier random forest classifier XGboost training set: 0.98 k-nearest classifier trai k-nearest classifier trai anaconda3\lib\site-packag ning: lbfgs failed to con . of ITERATIONS REACHED L umber of iterations (max_ ikit-learn.org/stable/mod fer to the documentation ikit-learn.org/stable/mod fer to the documentation ikit-learn.org/stable/mod fer to the documentation ikit-learn.org/stable/mod fer to the documentation ikit-learn.org/stable/mod g_msg=_LOGISTIC_SOLVER_CO logistic regression train logistic regression train logistic regression testi SVM training set: 0.95888 SVM testing set: 0.95630 Naive Bayes training set: Naive Bayes testing set:	0.99975 7533 training set: 0.99 testing set: 0.982 9975 254 ning set: 0.97723 ing set: 0.97723 es\sklearn\linear_l verge (status=1): IMIT. iter) or scale the ules/preprocessing for alternative so ules/linear_model. NVERGENCE_MSG) ing set: 0.96622 0.95888 0.96509 Python console History	975 54 model_log data as s .html lver optic html#logis	gistic.py shown in: ons: stic-regr	:940: ession			
	😧 conda: base (Python 3.7.6)	Line 115, Col 42	UTF-8	CRLF	RW	Mem	72%	

Fig 5.3: result without feature scaling

After that we apply the entire algorithms with feature extraction such as logistic regression, decision tree, XGBoost, Random Forest, KNN and then compare the result of all the algorithms. The 75% data used to train the model and remaining 25% data is used to test the model in all the algorithms.

Console 1/A ×					`∎ ⊿ ≡
After feature selection Accuracy on decision tree training set: Accuracy on decision tree test set: 0.97 Accuracy with random forest classifier of Accuracy with random forest classifier of Accuracy with XGboost training set: 0.98 Accuracy with XGboost testing set: 0.987 Accuracy with logistic regression training Accuracy with logistic regression testing Accuracy with logistic regression testing Accuracy with k-nearest classifier train Accuracy with k-nearest classifier test Accuracy with k-nearest classifier test Accuracy with SVM training set: 0.96053 Accuracy with SVM testing set: 0.96053 Accuracy with Naive Bayes training set: Accuracy with Naive Bayes testing set: 0 Accuracy with	0.99734 7571 training set: 0.99 testing set: 0.982 9734 140 ing set: 0.95711 ng set: 0.96471 ning set: 0.97887 ing set: 0.97647 0.95420 0.96053 0.95420 0.96053	734 92			
IF	ython console History				
😧 conda: base (Python 3.7.6)	Line 115, Col 42	UTF-8	CRLF	RW	Mem 80%

Fig 5.4: result with feature scaling

Chapter-6

RESULT AND PERFORMANCE ANALYSIS

Here we find the accuracy of different algorithms without feature extraction and then with feature extraction after that we compare the accuracy of both the algorithms. The algorithms are

- 1. Decision Tree
- 2. Random forest
- 3. XGBoost
- 4. K-nearest
- 5. Logistic Regression
- 6. SVM
- 7. Naïve Bayes

Now we show the accuracy of the algorithms without feature extraction on both training and testing set .

Algorithms	Accuracy			
	Training set	Testing Set		
Decision Tree	99.975	97.533		
Random forest	99.975	98.254		
XG Boost	99.975	98.254		
KNN	97.723	97.268		
Logistic Regression	96.179	96.622		
SVM	95.888	96.589		
Naïve Bayes	95.888	96.589		

Table 6.1: Results without feature extraction

Now we show the accuracy of the algorithms with feature extraction on both training and testing set.

Algorithms	Accuracy				
	Training set	Testing Set			
Decision Tree	99.734	97.571			
Random forest	99.734	98.292			
XG Boost	99.734	98.140			
KNN	97.887	97.647			
Logistic Regression	95.711	96.471			
SVM	95.420	96.053			
Naïve Bayes	95.420	96.053			

Table 6.2: Results with feature extraction

After comparing all the algorithms the algorithm **Random Forest** gives the most accurate result after features extraction that is **98.292%** .without feature extraction **Random Forest** and **XGBOOST** both gives the same result with highest accuracy that is **98.254%**.

Now we discuss the accuracy of each algorithm with their confusion matrix. First we display the confusion matrix without feature extraction and then with feature extraction. Confusion matrix has four tables. These sections are false negative and False Positive respectively and this occurs when there is a contradiction between actual class and the predicted class. The two sections are the True Positive and True Negative and these are the observations which are correctly predicted.

CONFUSION MATRIX WITOUT FEATURE EXTRACTION

1. Decision Tree

Activities			Fri Nov 29, 10:45 PM		4≬ ₽ -
					_ = ×
File Edit Search Source Run Debug Consoles Projects	Tools View Hel	0.		_	
		dec	ision_cm - NumPy array 🛛 🗖 🗖	× vents/malware-detection	
Editor - /nome/manas/Documents/mi/maiware_detection.py	0	1			© 18
□ ce.py X demographic.py X malware_detection.t	1709	36			w.
12	1708	30		Value	^
14 from sklearn, preprocessing import StandardSt	29	862		16J 86111	
15 sc=StandardScaler()				namos: Namo mds Machino SizoOfOnt	ional Header
16 X_train=sc.fit_transform(X_train)				26]	conacheader ,
17 X_test=sc.transform(X_test)				86211	
19 from sklearn.metrics import confusion matrix				36]	
20				863]]	
21 #decision tree				85511	
22 from sklearn.tree import DecisionTreeClassi				00011	
24 tree.fit(X train, v train)					
25 y_pred = tree.predict(X_test)					c x
<pre>26 decision_cm=confusion_matrix(y_test,y_pred)</pre>					= = =
27 print("Accuracy on decision tree training se				/34	■ <i>σ</i> •.
28 print(Accuracy on decision tree test set: 1					<u>^</u>
30 #random forest				ing set: 0.99734	
31 from sklearn.ensemble import RandomForestCl:				ng set: 0.98292	
32 classifier = RandomForestClassifier(n_estime					
33 classifier.fit(X_train, y_train)				at: 0.95711	
35 capdom cm=confusion matrix(v test v pred)				t: 0.96471	
36 print("Accuracy with random forest classific	Format	ze 🗸 Ba	ckground color	kages/sklearn/linear_model/logistic.	py:433:
37 print("Accuracy with random forest classifie				d to 'lbfgs' in 0.22. Specify a solv	er to silence
38			Save and Close Close		
39 #XGboost			Accuracy with k-pearest classifier train	100 set: 0 97887	
40 Trom XgDoost import AuBulassifier 41 yob-YGBClassifier(may depth-20 learning rate=0	n estimators-15	9)	Accuracy with k-nearest classifier testi	ing set: 0.97647	
42 xgb.fit(X train.v train)	, Cesterioterio IS	~,			
43 print("Accuracy with XGboost training set: {:.5	}".format(xgb.sco	ore(X_trai	In [2]:		
44 print("Accuracy with XGboost testing set: {:.5f	".format(xgb.sco	re(X_test,			*
45 y_pred = xgb.predict(X_test)			IPython console History log		
			Permissions: BW End of lines: LE	Encoding: ASCIL Line: 65 Column: 42	Momony 25%



Activities	Fr	i Nov 29, 10:45 PM	•1ù B →
			_ = ×
File Edit Search Source Run Debug Consoles Projects	Tools View Help		_
	random	n_cm - NumPy array 📃 🗖	× ents/malware-detection
Editor - /home/manas/Documents/ml/malware_detection.py			Ø X
c_ce.py X demographic.py X malware_detection.p	0 1		Q.
12 0	1731 13		Value
13 #feature scaling 14 from sklearn.preprocessing import StandardSc 15 scs-StandardScaler() 16 X_train=sc.fit_transform(X_train) 17 X totace_transform(X_text)	33 858		21] 823]] 13] 858]]
18 19 from sklearn.metrics import confusion_matrix			names: Machine, SizeOfOptionalHeader, Characteri [16] [17] [18] [19] [19] [19] [19] [19] [19] [19] [19
20 21 #decision tree 22 from sklearn.tree import DecisionTreeClassif 23 tree = DecisionTreeClassifier(random_state= 24 tree.fti(X_train, y_train) 25 w pred = tree.predict(X test)			861]] object of pandas.core.series module
26 decision_omeconfusion_natrix(y_test,y_pred) 27 print("Accuracy on decision tree training sr 28 print("Accuracy on decision tree test set: 39 arondom forest 31 from sklearn.ensemble import RandomForestCL 32 classifier = RandomForestClassifier(n_estime			734 ing set: 0.99734 ng set: 0.98292
33 classifier.fit(X_train, y_train) 34 y_pred = classifier.predict(X_test) 35 random_cm=confusion_matrix(y_test,y_pred) 36 print("Accuracy with random forest classifier 37 print("Accuracy with random forest classifier 38 difference 39 difference	Format Resize 🗹 Backgro	Save and Close	et: 0.95711 t: 0.96471 kages/sklarn/linear_model/logistic.py:433: d to 'lbfgs' in 0.22. Specify a solver to silence
40 from xgboot inport XGRClassifier 41 xgb-KGClassifier (max.depth-20,lerning_rate=0.3, 42 xgb.ft(X_train_y_train) 35 print("Accuracy with XGboot training set: {:.5f) 46 print("Accuracy with XGboot training set: {:.5f) 55 y ordet a xgb.redict(X test)	n_estimators=150) ".format(xgb.score(X_traiu .format(xgb.score(X_test,:	Accuracy with k-nearest classifier train Accuracy with k-nearest classifier testion In [2]:	nng set: 0.97687 ng set: 0.97647 ▼
4		Permissions: RW End-of-lines: LF	Encoding: ASCII Line: 65 Column: 43 Memory: 25%

2. <u>Random Forest</u>

Fig 6.2: random forest confusion matrix

3. XG Boost



Fig 6.3: XG Boost confusion matrix

4. <u>KNN</u>

Activities			Fri Nov 29, 10:46 PM			\$ G -
File Edit Search Source Run Debug Consoles Projects	Tools View H	eln		_		
		knea	rest_cm - NumPy array _		ents/malware-detection 👻	
Editor - /home/manas/Documents/ml/malware_detection.py						ØX
🗅 ce.py 🗶 demographic.py 🗶 malware_detection.p	0	1				٩,
40 from xgboost import XGBClassifier	1700	44			Value	
42 xpb.fit(X_train,y_train) 43 print(*Accuracy with XCboost training set: { 44 print(*Accuracy with XCboost testing set: { 45 y_pred = xxb_predict(X_test) 46 xpboost_cm=confusion_matrix(y_test,y_pred) 47 48 #Nakeing the confusion_matrix 49 from sklearn.netrics inport confusion_matrix 50 cn = confusion_matrix(y_test, y_pred) 51 52 #K-nearest neighbour 53 from sklearn.netrichbors inport KNeighborsClast 54 kclassifier-fit(X_train,y_train) 56 print(*Accuracy with k-nearest classifier the 59 xpared = kclassifier.predict(X_test) 59 knearest_cm=confusion_matrix(y_test,y_pred) 60 #JogisticRegression 62 from sklearn.linear_model import LogisticRef 63 Jr=LogisticRegression(random_state=2) 64 Jr.fit(X_train,y_train) 65 print(*Accuracy with logistic regression the 67 y_pred = kr.predict(X_test)	Format	size V Back	kground color	Close	<pre>essip 261 263 264 264 264 265 264 264 264 274 274 274 274 274 274 274 274 274 27</pre>	e e e e e e e e e e e e e e e e e e e
68 log_cm=confusion_matrix(y_test,y_pred) 69 ''' 70 #svm 71 from sklappo sym import SVC			Accuracy with k-nearest classifie	er testing s	et: 0.97647	
72 svm_classifier.fu(kernel='linear') 73 svm_classifier.fit(X_train,y_train)			IPuthen concolo History log			•
•		p	Permissions: PW End-of-lin	nes IE En	coding: ASCII Line: 65 Column: 43 Memo	25 %

Fig 6.4: KNN confusion matrix

5. Logistic Regression

Activities	Fri Nov 29, 10:46 PM	•0, ₽ -
		_ = ×
File Edit Search Source Run Debug Consoles Project	ts Tools View Help	
🗋 🖕 🖹 🐂 🧰 🖉 🕨 🚺	log_cm - NumPy array	- 🔍 × nents/malware-detection 🔹 🍃 🛧
Editor - /home/manas/Documents/ml/malware_detection.py		@ 8
C_ce.py X demographic.py X malware_detection.c	0 1	Q.
43 print("Accuracy with XGboost training set:	0 1723 21	Value
<pre>types - with ext(t(lest)) types - with ext(t(lest)) types - with ext(t(lest)) types - with ext(lest) types -</pre>	Format Resize Background color	dij di di
6) yore - (r,pleat(r,_test) 6) log_confusion_matrix(ytest,y_pred) 6) ''' 70 #svm 71 from sklearn.svm inport SVC 72 svm_classifier_fit(krenel=') 73 svm_classifier_fit(krenel=')	Save and Close Accuracy with k-nearest classif Accuracy with k-nearest classif	d to 'lbfgs' in 0.22. Specify a solver to silence ter training set: 0.97897 ter testing set: 0.97647
74 print("Accuracy with SVM training set: {:.5f}" 75 print("Accuracy with SVM testing set: {:.5f}". 76 '''	.format(svm_classifier.score; In [2]: format(svm_classifier.score(; Putton console History Ion	v
	Permissions: RW End-of	-lines: LF Encoding: ASCII Line: 65 Column: 43 Memory: 25%





Fig 6.6 : SVM confusion matrix

7. GaussianNB



Fig 6.7: Gaussian Naïve Bayes confusion matrix

CONFUSION MATRIX WITH FEATURE EXTRACTION

1. Decision Tree

Activities	Fri Nov 29, 10:50 PM	•↓ 🔒 -
	Spyder (Python 3.7)	_ = ×
File Edit Search Source Run Debug Consoles Project	Tools View Help	
	fdecision_cm - NumPy array 🛛 🗕 🗙	ents/malware-detection - 🗲 🛧
Editor - /home/manas/Documents/ml/malware detection.py		ØX
Co ce.pv X demographic.pv X malware detection.c	0 1	۵.
90 select=RFE(ExtraTreesClassifier(n estimators	1708 36	Value
91 select.fit(X_train,y_train) 92 Xt_mewselect.transform(X_train) 93 Ytest_newselect.transform(X_test) 94 ''' 95 print("After feature selection") 96 #decision tree 97 from sklearn.tree import DecisionTreeClassif 98 tree = DecisionTreeClassifier(random, state= 99 tree.fit(Xt_new, y_train) 100 print("Accuracy on decision tree training of 101 print("Accuracy on decision tree training of 102 y_pred = tree.predict(Xtest_new) 103 fdecision_cneconfusion_matrix(y_test,y_pred) 104 105 from sklearn.ensemble import RandonForestCL 107 classifier = RandonForest Classifier("n_estimations forest 118 print("Accuracy with random forest classifier 119 y_pred = classifier.predict(Xtest_mew) 119 print("Accuracy with random forest classifier 111 yprint("Accuracy with random forest classifier 111 yprint("Accuracy with random forest classifier 113 print("Accuracy with random forest classifier 114 print("Accuracy with random forest classifier 115 print("Accuracy with random forest classifier 116 from sphoost import XCBClassifier 117 xyb=XCBClassifier(max_deptin=20,learning_rature) 118 print("Accuracy with XCBOOst training set: [:.5 120 print("Accuracy with XCBOOst training set: [:.5 120 print("Accuracy with XCBOOst testing set: [:.5] 121 print("Accuracy with XCBOOst testing set:	28 863 Format Resize V Background color Save and Close Close P*.format(xgb.score(Xt_new *.format(xgb.score(Xtest_new) Accuracy with k-nearest classifier training trai	names: Name, md5, Machine, SizeOfOptionalHeader,- 361 362] 363] 263] 263] 264] 264] 264] 264] 264] 264] 264] 267 268 264] 268 264] 268 264] 268 264] 268 268 268 268 268 268 268 268
122 fxgboost_cm=confusion_matrix(y_test,y_pred)	· · · · · · · · · · · · · · · · · · ·	*
€	Permissions: PW Endofilines: LE Fr	ronding: ASCII Line: 65 Column: 43 Memory: 25%



2. <u>Random Forest</u>

Activities			Fri Nov 29, 10:51 PM			4% ₽ -
						_ • ×
File Edit Search Source Run Debug Consoles Prov	acts Tools View	Help				
🗋 🖕 🖹 🐂 🧮 🖉 🕨 🕻		frand	om_cm - NumPy array		ents/malware-detection	- 5 1
Editor - /home/manas/Documents/ml/malware_detection.py	-					0 1
🔁 çe.py 🗶 demographic.py 🗶 malware_detection.p	0	1				Q.
<pre>35 print("After feature selection") 36 daccision tree 37 from sklearn.tree import DecisionTreeClassi 38 tree = DecisionTreeClassifier(random_statest 39 tree.fl(XL]new, yrtan) 100 print("Accuracy on decision tree training se 101 print("Accuracy on decision tree test set: 102 print accuracy on decision tree test set: 103 decision_cm-confusion_matrix(y_test,y_pred) 104 decision_cm-confusion_matrix(y_test,y_pred) 105 from sklearn.ensemble import RandomForestCl 107 classifier = RandomForestClassifier(n_estim 108 classifier = RandomForest classifier 110 print("Accuracy with random forest classifier 111 print("Accuracy with random forest classifier 112 spred = classifier.predict(Xtest,new) 113 frandom_cm-confusion_matrix(y_test,y_pred) 114 115 #Xcboost 116 from xgboost import XGBClassifier 117 xgb=XGBClassifier(nx_depth=20,learning_rate 118 xgb.f(Xt_new,y_train) 119 print("Accuracy with XGboost training set: 121 y.pred = xgb.predict(Xtest,new) 122 forgboost_cm-confusion_matrix(y_test,y_pred) 123 124 deligitic repression</pre>	0 1726 1 27 Format 1	18 864 tesize V Back	ground color Save an Accuracy with k-nearest c Accuracy with k-nearest c	d Close Close	Value 36] 36] 26] 26] 26] 26] 26] 26] 26] 2	istic.py:433: a solver to silence
126 lr=LogisticRegression(random_state=2) 127 lr.fit(Xt_new,y_train) 128 print("Accuracy with logistic regression trai	ning set: {:.5f}	.format(lr.s	In [2]:	3		*
[4]		•	Permissions: RW	End-of-lines: LF Er	coding: ASCII Line: 65 Colu	imn: 43 Memory: 25%

Fig 6.9: After feature extraction Random Forest confusion matrix

3. XG Boost

Activities	Fri Nov 29, 10:51 PM	•@ € -
		_ = ×
File Edit Search Source Run Debug Consoles Projects	: Tools View Help	
🗅 🖕 🖺 🐁 🗮 @ 🕨 🖪 🖬 🚺	fxgboost_cm - NumPy array _ 🗖 🗙	ents/malware-detection 🔹 🗲 🛧
Editor - /home/manas/Documents/ml/malware_detection.py	0 1 1723 21	© 8 ¢. Value
<pre>39 tree = Dectsion/reeClassifier(random_statest) 9 tree.fit(X1_new, y_train) 1 100 print('Accuracy on decision tree test set) 101 print('Accuracy on decision tree test set) 102 grand testing and the decision tree test set) 103 grandom forest (lassifier(n_testing)) 104 grandom forest (lassifier(n_testing)) 105 grandom forest (lassifier(n_testing)) 105 grandom forest (lassifier(n_testing)) 106 print('Accuracy with random forest classifier(n_testing)) 107 grandom_cn=confusion_natrix(y_test,y_pred) 108 frandom_cn=confusion_natrix(y_test,y_pred) 104 frandom_cn=confusion_natrix(y_test,y_pred) 104 frandom_cn=confusion_natrix(y_test,y_pred) 105 from stybost import XGBClassifier 107 xgb=XGBCost 108 print('Accuracy with Xboost training_rate 109 print('Accuracy with Xboost training_rate 109 print('Accuracy with Xboost training_set: { 100 print('Accuracy with Xboost testing set: { 100 print('Accuracy with Xboost testing set: { 100 print('Accuracy with Xboost testing_set: { 100 print('Accuracy with Xb</pre>	28 863	36] Value 36] 263] 263] 263] 263] 263] 824] 864] 864] 864] 21] 864] 863] 98292 rt: 0.98734 10 ng set: 0.98292 1 rt: 0.98711 1
121 y_pred = xgb.predict(Xtest_new) 122 fxgboost_cm=confusion_matrix(y_test,y_pred) 123 124 #logistic regression 125 from sklear_linear model import lonisticRec	Format Resize V Background color	<pre>kapes/sklearn/linear_model/logistic.py:433: d to 'lbfgs' in 0.22. Specify a solver to silence</pre>
<pre>126 lr=logisticRegression(random_state=2) 127 lr.fit(X1_new,y_train) 128 print("Accuracy with logistic regression traini 129 print("Accuracy with logistic regression testing 130 y_pred = lr.predict(Xtest_new) 131 flog_meconfusion_matrix(y_test_y_pred)</pre>	ng set: {:.5f}".format(lr.s. g set: {:.5f}".format(lr.s.	ng set: 0.97887 ; set: 0.97647 ▼
4	IPython console History log	

Fig 6.10: After feature extraction XGBoost confusion matrix

4. Logistic Regression

Activities	Fri Nov 29, 10:51 PM	🚸 B 🗸
		_ = ×
File Edit Search Source Run Debug Consoles Projects	Tools View Help	
🗋 🖕 🖺 🐁 🧮 @ 🕨 🛃 🔂 🗘 🤇	flog_cm - NumPy array _	• × vents/malware-detection • ► 个
Editor - /home/manas/Documents/ml/malware_detection.py		@ 8
C_ce.py X demographic.py X malware_detection.c	0 1	0,
<pre>18% classifier.fit(Xt_new, y_train) 18% classifier.fit(Xt_new, y_train) 180 print('Accuracy with random forest classifier 112 y_pred = classifier.predict(Xtest_new) 113 frandom_emeconfusion_matrix(y_test,y_pred) 114 115 #Xkboost 116 from xgboost inport XGBClassifier 117 xgb=XGBClassifier(ng, dgbh=20, learning_rate 118 xgb.fit(Xt_new, y_train) 119 print('Accuracy with XGboost training set: 120 print('Accuracy with XGboost training set: 121 y_pred = xgb.predict(Xtest_new) 122 fogboost_emeconfusion_matrix(y_test,y_pred) 123 124 flogistic regression 125 from sklearn.linear_model inport LogisticRes 128 uprint('Accuracy with logistic regression tr 129 print('Accuracy with logistic regression tr 129 print('Accuracy with logistic regression tr 129 print('Accuracy with logistic regression tr 129 uprint('Accuracy with logistic regression tr 120 uprint('Accuracy with logistic regression tr 120 uprint('Accuracy with logistic regression tr 120 uprint('Accuracy w</pre>	1718 26 67 824	36) 36) 863]] 26] 863]] 26] 863]] 863]] 864]] 21] 863]] 863]] 863]] 864]] 21] 863] 863] 863] 863] 863] 863] 863] 863] 863] 863] <
<pre>138 print("Accuracy with k-nearest classifier testing 139 y_pred = kclassifier.predict(Xtest_new) 140 fknearest_cm=confusion_matrix(y_test,y_pred)</pre>	set: (:.5f) .format(kcla In [2]:	v
•	IPython console History log Permissions: RW End-of-line	es: LF Encodina: ASCII Line: 65 Column: 43 Memory: 25%

Fig 6.11: After feature extraction logistic regression confusion matrix

5. <u>KNN</u>

Activities	Fr	i Nov 29, 10:52 PM			•% ₽ -
	S	pyder (Python 3.7)		-	. • ×
File Edit Search Source Run Debug Consoles Proj	acts Tools View Help		_		_
🗋 🖕 🖹 🐂 🧮 @ 🕨 🖬 🛃 🗣 🤇	fkneare	st_cm - NumPy array _		ents/malware-detection	
Editor - /home/manas/Documents/ml/malware_detection.py					ØX
cce.py X demographic.py X malware_detection.p	0 1				Φ,
<pre>188 classifier.fit(Xt_new, y_train) 189 classifier.fit(Xt_new, y_train) 189 y_pred = classifier.predict(Xtest_new) 181 print("Accuracy with random forest classifier 181 print("Accuracy with Xcbost resting") 185 arAchoost 185 arAchoost 186 from xgboost inport XCBClassifier 185 print("Accuracy with XCbost resting set: 180 print("Accuracy with XCbost resting set: 180 print("Accuracy with XCbost resting set: 182 print("Accuracy with logistic regression 182 from sklearn.linear_model import LogisticRev 183 print("Accuracy with logistic regression test 183 print("Accuracy with logistic regression 183 from sklearn.neighbor 184 from sklearn.neighbors(lassifier("n.neighbors(lassifier.Nit("Accuracy with k-nearest classifier test 185 print("Accuracy with k-nea</pre>	0 1718 26 1 36 855 Format Resize V Backgr thing set: {:.5f}*.format(kcl. titing set: {:.5f}*.format(kcl.	Accuracy with k-nearest classifier In [2]:	Close training s	Value 36) Value 36) 36) 36) 36) 36) 36) 36) 36)	a a a a a a a a a a a a a a a a a a a
concerned to concern the concerned to concern the concerned of the concern		Python console History log			•
	•	Permissions: RW End-of-lines	s: LF En	coding: ASCII Line: 65 Column: 43 Mem	ory: 25%

Fig 6.12: After feature extraction KNN confusion matrix

6. <u>SVM</u>

Spyder (Python 3.7)		- 0 ×
File Edit Search Source Run Debug Consoles Projects Tools View	Help	
🗅 늘 🖻 🐂 🗮 @ 🕨 📮 🛃 🕩 🧲	州 📫 🔚 🚛 🍌 📕 🔝 🔀 🎤 🥏 🔶 🔶 C:Libers'aceriuspyder-py3	1 🖙 🛧
C:\Users\acer\.spyder-py3\temp.py		=
temp.py ×	tsvm_cm - NumPy array — 🗆 X	
53 xeboost cm=confusion matrix(v test,v pred)		Size Value
54	0 1	(2, 2) [27 864]]
55 #Makeing the confusion matrix 56 from sklearn metrics import confusion matri	0 1718 26	(2, 2) [[1718 26]
57 cm = confusion_matrix(y_test, y_pred)	1 36 855	
58		(2, 2) [28 863]]
59 #K-nearest neighbour 60 from sklearn.neighbors import KNeighborsCla	n.KNeighborsClassifier	1 KNeighborsCla…
61 kclassifier=KNeighborsClassifier(n_neighbor		(2 2) [[1700 44]
62 kclassifier.fit(X_train,y_train) 63 print("Accuracy with b-menest classifier t		[28 863]]
64 print("Accuracy with k-nearest classifier t		(2, 2) $[68 823]]$
65 y_pred = kclassifier.predict(X_test)	pristicRepression	1 LogisticRegre
67	volorer Help Plots Files	
68 #logistic regression		·
69 from sklearn.linear_model import LogisticRe		
71 lr.fit(X_train,y_train)	1.99/34	<u>_</u>
72 print("Accuracy with logistic regression tr	'/' 'aining set: 0.99734	
<pre>73 print("Accuracy with logistic regression te 74 v pred = lr.predict(X test)</pre>	sting set: 0.98292	
<pre>75 log_cm=confusion_matrix(y_test,y_pred)</pre>	'34 IA	
76	ig set: 0.95711	
77 #SVM 78 from sklearn.svm import SVC	; set: 0.96471	
<pre>79 svm_classifier=SVC(kernel='linear')</pre>	ing set: 0.97647	
80 svm_classifier.fit(X_train,y_train) 81 print("Accuracy with SVM training set: /: 5	Format Resize Recipe Record color	
82 print("Accuracy with SVM testing set: {:.5		
<pre>83 y_pred = lr.predict(X_test)</pre>	Save and Close Close 96053	
84 svm_cm=confusion_matrix(y_test,y_pred) 85),95420	
86 #Naive Bayes	Accuracy with Naive Bayes testing set: 0.96053	
87 from sklearn.naive_bayes import GaussianNB		Ų
	IPuthen concole History	
t	G conda: base (Python 3.7.6) Line 197, Col 1 UTF-8	CRLF RW Mem 83%

Fig 6.13: After feature extraction SVM confusion matrix

7. <u>GaussianNB</u>

😵 Spyder (Python 3.7)			– 0 ×
File Edit S	earch Source Run Debug Consoles Projects Tools Vie	Help		
🗅 🍃	🖪 🛼 🔳 @ 🕨 🖻 🛃 📭 🧲) 🕻 🔚 🚛))) 📘 🖸 💱 🎤 🍦 🗲 🄶 C:(Userslacer).spyder-py3		* 📥
C:\Users\ace	r\.spyder-py3\temp.py	fey an - NumPy stay		≡
🖪 temp.p	у ×			
98	ny fit(X train y train)	· •	Size	Value
99	<pre>print("Accuracy with Naive Bayes training :</pre>			[[1/18 26]
100	print("Accuracy with Naive Bayes testing s	0 1718 26		[[1726 18]
101	<pre>y_pred = lr.predict(X_test)</pre>		(2, 2)	[27 864]]
102	<pre>nv1_cm=confusion_matrix(y_test,y_pred)</pre>	1 36 855	(2, 2)	[[1718 26]
103	#with feature selection			[36 855]]
105	from sklearn.feature selection import Selection			[[1/23 21]
106	from sklearn.ensemble import ExtraTreesCla			[20 000]]
107	<pre>select=SelectFromModel(ExtraTreesClassifie</pre>	h.KNeighborsClassifie		KNeighborsCla
108	select.fit(X_train,y_train)			[[1700 44]
110	Xt_new=select.transform(X_train) Xtest_new=select.transform(X_test)		(-) -/	
111	xcest_new=sereeeeen ansion m(x_cest)		(2, 2)	
112		Explorer Help Plots Files		
113	#feature selection using recursive feature			
115	from sklearn.feature selection import RFE			
116	from sklearn.ensemble import ExtraTreesCla	96509		
117	select=RFE(ExtraTreesClassifier(n_estimato			
118	select.fit(X_train,y_train)	.99734		
119	Xt_new=select.transform(X_train)	71		
120	<pre>/// // // // // // // // // // // // //</pre>	aining set: 0.99734		
122	print("After feature selection")	sting set: 0.98292		
123	#decision tree	2* A		
124	from sklearn.tree import DecisionTreeClass:	g set: 0.95711		
125	<pre>tree = DecisionTreeClassifier(random_state</pre>	set: 0.96471		
126	tree.fit(Xt_new, y_train)	Format Resize Background color .ng set: 0.97887		
127	print("Accuracy on decision tree test set:	g set: 0.97647		
129	y pred = tree.predict(Xtest new)	Save and Close Close		
130	fdecision_cm=confusion_matrix(y_test,y_pred	Accuracy with Naive Bayes training set; 0.95420		
131		Accuracy with Naive Bayes testing set: 0.96053		
132	#random forest	Accuracy with Naive Bayes training set: 0.95420		
133	Trom Skiearn.ensemble import RandomForestC.			
		IPython console History		
-				

Fig 6.14: After feature extraction Gaussian Naïve Bayes confusion matrix

<u>Chapter-7</u> <u>CONCLUSION&FUTURE SCOPE</u>

CONCLUSION

Various ML algorithms of supervised learning has been applied for the detection and removal of infected malware files or anomaly in the given dataset samples and classified them into 2 categories that is malicious and benign. We used a dataset that was taken from kaggle. We were able to get a labeled dataset and then applied various supervised ML algorithms and split the dataset into 75% training and 25% testing set. With this dataset, we evaluated the values of different accuracies of algorithms and thus evaluated the parameters and features for various supervised algorithms and concluded that the machine learning algorithm which gave the best accuracy is the Random Forest. Thus with an accuracy of 98.29% after feature extraction and 98.25% before feature extraction, it is the best algorithm for this dataset.

FUTURE SCOPE

For the future work we will implement deep learning and use neural networks for the recognition of rare malwares and we will also run a live malware file on a virtual environment like Anubis and then get its log file for filling the entries in the data set and hen using the predefined machine learning algorithm for predicting whether the file is malicious or not. And with more features and parameters of algorithms added to increase the accuracy of performance matrix. Evaluation of malwares only on the basis of static approach also can be used like prediction of malware on the basis of analysis of code and signatures.

REFERENCES

- [1] Dataflair Team. (2020, January) www.data-flair.training. [Online]. <u>https://data-flair.training/blogs/machine-learning-tutorial/</u>
- [2] (2013) www.analyticsvidhya.com. [Online]. <u>https://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2016/03/cost1.png?resize=564%2C232</u>
- [3] Kambria Team. (2019, July) www.kambria.io. [Online]. <u>https://kambria.io/blog/confused-about-the-confusion-matrix-learn-all-about-it/</u>
- [4] Purplesec Team. (2019) www.purplesec.us. [Online]. <u>https://purplesec.us/resources/cyber-security-statistics/</u>
- [5] Ekta Gandotra, Divya Bansal, Sanjeev Sofat. (2014, April) www.scrip.org. [Online]. https://file.scirp.org/pdf/JIS_2014040110394271.pdf
- [6] Ivan Firdausi, Charles Lim. (2010, December) www.ieeexplore.ieee.org. [Online]. <u>http://ieeexplore.ieee.org/iel5/5672650/5675795/05675808.pdf</u>
- [7] Simran Kaur Arora. (2020, April) hackr.io. [Online]. <u>https://hackr.io/blog/supervised-vs-unsupervised-learning</u>
- [8] Adebayo Segun. (2019, August) www.researchgate.net. [Online]. https://www.researchgate.net/figure/Support-Vector-Machine fig1 336020465
- [9] Saimadhu Polamuri. (2017, March) dataaspirant.com. [Online]. <u>https://dataaspirant.com/how-logistic-regression-model-works/</u>

PLAGIARISM REPORT

Proj	ect report				
ORIGIN	ALITY REPORT				
9 SIMILA	% RITY INDEX	1% INTERNET SOURCES	7% PUBLICATIONS	4% STUDENT P	PAPERS
PRIMAR	Y SOURCES				
1	Aditya Ka Gandotra Applicatio 2019 Inte Processin Publication	poor, Himanshu . "Permission ba on Detection usir rnational Confer ng and Commun	Kushwaha, E nsed Android M ng Machine Le rence on Signa ication (ICSC)	kta Malicious earning", al), 2019	3%
2	Firdausi, Satriyo N Techniqu Detection Conferen and Teleo Publication	Ivan, Charles lin ugroho. "Analysi es Used in Beha ", 2010 Second ce on Advances communication T	n, Alva Erwin, is of Machine avior-Based M International in Computing Fechnologies,	and Anto learning alware Control 2010.	2%
3	Submittee Student Paper	to The Robert	Gordon Unive	rsity	1%
4	Submitted Student Paper	d to University of	f Strathclyde		<1%
5	www.ijert	.org			<1%

Hongtai Yang, Xiaozhao Lu, Christopher Cherry,