Artificial Intelligence enabled Framework for Classification of Windows Portable Executable Malware using Static Feature Analysis

By

WARDA ASLAM

00000278053

Supervisor

Dr. MUHAMMAD MOAZAM FRAZ

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science (MSCS)

In

School of Electrical Engineering and Computer Science

National University of Sciences and Technology(NUST)

Islamabad, Pakistan

August 2020

Approval

It is certified that the contents and form of the thesis entitled "AI enabled framework for classification of Windows portable executable malware using static feature analysis" submitted by WARDA ASLAM have been found satisfactory for the requirement of the degree

Advisor: **Dr. Muhammad Moazam Fraz**

Signature: [illegible]

Date: 27-July-2020

Committee member-1: **Dr. Muhammad Khurram Shahzad**

Signature: [illegible]

Date: 27-Jul-2020

Committee Member 2: **Dr. Muhammad Shahzad**

Signature: M. SHAHZAD

Date 27-Jul-2020

Committee Member 3: **Dr. Arsalan Ahmad**

Signature: [illegible]

Date: 28-Jul-2020

Dedication

I dedicate this thesis to my inspirations in life, my FATHER MUHAMMAD ASLAM, who taught me how to fight against the struggles of life, my MOTHER (late) who always wanted to study to every extent possible, in order to fulfill her wish I definitely dedicate this thesis work to her.

Certificate of Originality

I hereby declare that this submission titled "AI enabled framework for classification of Windows portable executable malware using static feature analysis" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student name: Warda Aslam

Signature: [illegible]

ACKNOWLEDGEMENT

First and foremost, I am exceptionally grateful to Allah Almighty Without whose help this thesis work couldn’t have been completed. All the help, guidance and support from my parents and teachers were all because of Allah’s will.

I am grateful to my supervisor, Assistant Professor, Dr. MUHAMMAD MOAZAM FRAZ, for the tremendous supervision, motivation, and guidance he has conveyed all through my time as his understudy. I have been exceptionally honored to have a supervisor who thought such a great amount about my work, and who acknowledged to my requests and questions immediately.

I am especially appreciative to my father who supported me throughout my life, who steadfast with me in all circumstances I had to face during this thesis work, I am grateful to him for providing every possible type of support

Thank you

WARDA ASLAM

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI: Artificial Intelligence

AE: AutoEncoder

AUC: Area under Curve

DNN: Deep Neural Network

FN: False Negative

FP: False Positive

FPR: False Positive Rate

IP: Internet Protocol

KNN: K- Nearest Neighbors

ML: Machine Learning

MS: Microsoft

NN: Neural Network

PC: Personal Computer

PCA: Principal Component Analysis

PE: Portable Executable

RF: Random Forest

ROC: Receiver Operating Characteristics

SVM: Support Vector Machines

TP: True Positive

TN: True Negative

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Keeping your computer system safe from all types of viruses, trojans, spyware, Ransomes is a daily basis struggle. All around the World people are struck by this problem on a daily basis. Using anti-virus is so far the best possible cure found till now. The problem with anti-virus is that it is unable to detect any new type of malware; it will traditionally match the characteristics of the previously detected malware with the newly detected malware. It can easily be fooled by malware with different characteristics and hence your system gets infected. To overcome this hurdle artificial learning approach is applied for this thesis work. Machine learning has tremendous power to predict based on training done previously. One essential for artificial intelligence is large amounts of datasets. One of the goals of this research work was to collect enough dataset to apply machine learning. Only static features were drawn out from benign and malware PE files for classification. Two datasets were used a publicly available dataset and a self-collected dataset of about 21,000 samples. In machine learning, unsupervised algorithms using the resultant features given by PCA gave precision and recall above 0.8%. Results produced by machine learning supervised and unsupervised algorithms resulted in above 80% training and testing accuracy. Best results were given by dimensionality reduction approaches. Above 90% accuracy was achieved in proposed dimensionality reduction models. This approach was pronounced to be better than the traditional signature-based malware detection techniques due to its ability to learn and predict.

Keywords

Malware, artificial intelligence

.

CHAPTER 1

INTRODUCTION

1.1 Malware: A threat across Globe

‘Malicious software is in short called malware. Malware comprises programming designed to gain unauthorized access to system resources, gather the information that leads to loss of privacy or exploitation, disrupt or deny operation and other abusive behavior’ [1]. Every country is affected by malware. Malware creators create malware to perform a specific operation. The extent of malware damage can be determined based upon the target of the malware. Being a home user, a virus may not do major damage, in case a malware enters a corporate organization or any network it can gain access to confidential information and can cause way more damage than that caused to a home user.

Malware can easily enter your computer system if you download executable files from any unauthentic websites it can transfer malware into your computer with the executable. Malware can also get into your computer if you click on any dubious link or any email from an unknown or unauthentic email address. Any removable device is also able to carry malware.

In February 2020, a management company based in Denmark was struck by a cyber-attack of prime importance security attacks. All of the computer systems were shut down with many operations on a halt. Later on, many connected computer systems were restored, but nearly 43,000 employees were still not able to access their emails or any other online access [2].

Nearly one-third of computer systems are globally affected by malware attacks annually [3]. For the first time in 2016, lower rates of new malware samples were reported to be of 127 million. Surprisingly a malware of entirely new characteristics emerged 4.2 seconds in 2017 and in 4.6 seconds in 2016 [4].

Pakistan stands at a high risk of malware attacks as reported by Microsoft in 2017. According to Microsoft report Pakistan, Bangladesh, Egypt and Indonesia is expected to suffer from malware attacks most in the years to come [5].

According to one of the survey every one out of four computer systems in Pakistan is infected at some time by malware. Also, it was reported that nearly 84% of the computer systems in Pakistan don't use any security measure against malware attacks [6].

Malware can be detected using anti-virus, yet malware can by-pass anti-virus and can damage your computer system or especially your network to any extent a malware is capable of. Preferably one should protect their computer systems to every extent possible. The best way to keep your computer systems safe is to keep the anti-virus up to date and other software updated.

The reason why malware can get pass that anti-virus software is that traditionally signature-based anti-virus software is used. Signature-based anti-virus software is incapable of detecting a malware of slightly different characteristics than the one previously detected.

Signature-based ant-virus keeps track of all the previously encountered malware and once a new malware tries to enter the user's computer system, it matches characteristics of the new malware with those previously stored on the anti-virus database. Signature-based anti-virus software will match the characteristics, it will check upon characteristics of the family of malware and in the end it can check upon the suspicious structure of the malware executable. Malware with a few changed properties can easily get past signature-based anti-virus.

1.2 Problem Statement

The basic problem statement around which this thesis work evolves is "to collect enough dataset to apply artificial intelligence techniques to overcome traditional signature-based techniques". For the training phase of the model large amount of dataset is needed. Therefore one of the basic goals of this thesis work is to collect enough samples for training and testing.

1.3 Anti-virus verses Anti-malware

To protect one's computer system usually most common searches comprises of 'best anti-virus' or 'latest anti-virus'. What's important here is that there is a difference between viruses and malware. A virus is a snippet of code efficient enough to duplicate itself once it enters users' computer systems. A virus can replicate itself to cause harm to your machine or PC. A virus can corrupt your data as well as your system [7].

On the other hand, malware can be considered an aggregation of all the software deliberately created to cause damage to computer systems. Malware includes spyware, Trojans, worms, etc.

Anti-virus will try to check all types of malware including worms, Trojans etc. This is the reason why anti-virus is more commonly installed to protect the computer system. What makes an anti-malware better than anti-virus software is their job to detect the polymorphic nature of viruses. A polymorphic virus is a virus with somewhat different characteristics than the usual malware.

Some of the most updated and latest anti-virus software includes McAfee [8], TotalAV [9], Norton [10], and many others are available. One of the anti-malware is MalwareBytes [11]. It is better recommended by professionals to use both types of software that is anti-virus software and anti-malware software.

1.4 Justification of research

As stated in problem identification, traditionally signature-based anti-virus is used. Such techniques can be fooled quite easily if the malware is created with some trick. The creators of malware can change the signature of the virus being created so that it cannot be perceived by the anti-virus.

A signature-based anti-virus works with already known signatures or byte patterns. A hacker can easily just change the byte pattern or perform obfuscation so that signature-based anti-virus software doesn’t have that byte pattern stored in its database. Malware code can easily be changed and it can bypass anti-virus detection mechanisms [12].

Another problem that traditional signature-based anti-virus software suffers from is the problem of zero-day attacks. ‘A zero-day attack will check for a short-coming of the system whether it be logical or time-based, it will attack taking the advantage of some shortcoming faced by the system’ [13]. A zero-day attack is one of the most different and difficult types of malware’s to be detected and then also removing such malware demands a lot of effort and experience in this field. A zero-day attack is one of its kinds. It can cause major damage due to its nature of entirely different characteristics and nature. It may have entirely different characteristics than any previous malware attacks.

How a zero-day work is important. A zero-day attack will be created based upon a deficiency in the system or based upon any event that has to take place soon. It is called zero-day as it is just created and is about to be deployed or is already deployed for the first time on the targeted computer system or computer systems.

The problem arises here for traditional signature-based anti-virus malware because they are unaware of any new type of malware. Zero-day attacks are polymorphic in nature and such characteristics are not known by signature-based anti-virus software. Once a zero-day attack is deployed it can cause enough damage already before a remedy is launched for the attack.

To overcome such signature-based techniques artificial intelligence-based techniques can overcome these problems to some extent. The property of machine learning to train and learn from different samples makes it capable to predict in real-time based upon training. More the training better the model prediction accuracy.

Similarly, deep learning holds even more power than machine learning. Both techniques are data and training dependent. More the data for training the better the malware detection rate in real-time. Deep learning will be able to detect the polymorphic nature of the newly detected malware. Therefore, both the techniques are capable of overcoming the problem of signature-based malware detection techniques.

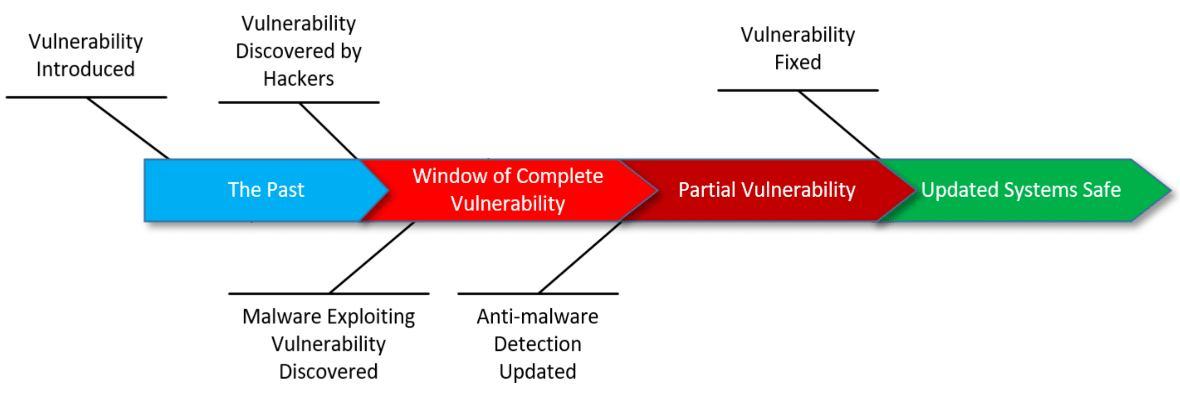

Vulnerability Introduced
Vulnerability Discovered by Hackers
Vulnerability Fixed
The Past
Window of Complete Vulnerability
Partial Vulnerability
Updated Systems Safe
Malware Exploiting Vulnerability Discovered
Anti-malware Detection Updated

Fig 1. 1: A zero-day attack time line [1].

1.5 Research questions

- How can artificial intelligence based techniques overcome signature-based techniques?

This above question is divided into sub questions:

- Which machine learning algorithms to use?
- Which deep learning techniques to be used?
- How will malware detection accuracy be increased in real-time?
- How much dataset to be used for good training results?
- How to gather dataset to apply the proposed approach?

1.6 Objectives of Research

The foremost objectives of this research are:

1. To collect dataset
2. Validate self-collected dataset
3. To create a robust model
4. To propose/improve models for clustering
5. To compare different feature dimensionality reduction techniques and find out which suits best for clustering
6. To validate the results using quantitative and evaluation metrics.

1.7 Research Contribution

1) Self-collected dataset of 21,486 portable executable files.
2) Robust model using cross validation using two datasets one which was self-collected and other one that was accessed from web (publicly available since 2018)
3) Utilization of GMM as clustering algorithm (GMM is used as feature selector in literature).
4) Fuzzy c-means introduced for benign-malware clustering.
5) A model proposed for clustering based upon dimensionality reduction techniques crosses results of a similar technique seen in literature with about difference of 3-4% in F1- score as well as better precision and recall.

6) Better clustering models proposed with resultant precision, recall and F1-score between 0.8-0.95.

1.8 Organization of Thesis

- Chapter 1, Introduction

  Malware: A threat across Globe, Problem identification, Anti-virus verses anti-malware, Zero-day attacks, Justification of research, Objectives

- Chapter 2, Literature review

  Types of malware analysis, related work, Portable executable

- Chapter 3, Data acquisition

  Publicly available dataset, self-collected dataset, features used in dataset, comparison of benign and malware threshold values

- Chapter 4, Methodology

  Basic flow: methodology part-1, goal: a robust model, strategy for supervised algorithms, strategy for unsupervised algorithms, Basic flow: Methodology Part-2, clustering based upon static features, simple clustering, clustering with dimensionality reduction, clustering with ensemble NN based autoencoder.

- Chapter 5, Results and Discussion

  Parameters for supervised algorithms, numerical results for supervised machine learning algorithms, graphical results for supervised machine learning algorithms, classification reports for supervised algorithms, ROC curve for supervised machine learning algorithms, numerical results for un-supervised algorithms, result analysis of machine learning algorithms, clustering results for static features with dimensionality reduction

- Chapter 6,

  Conclusion

Summary

The different aspects related to malware and why malware detection is important is discussed in this chapter. Several malware attacks have been discussed also the situation of Pakistan in terms of malware threats. Why to use anti-malware instead of anti-virus is discussed in this chapter, also the justification for this research topic is discussed. Moreover, the research objectives are also stated while at the end, thesis organization has been

CHAPTER 2

LITERATURE REVIEW

2.1 Types of malware analysis

‘Malware analysis is the procedure of figuring out how malware functions in different scenarios’ [3]. In malware analysis, the analyzer is capable of analyzing and understanding the basic characteristics of malware or benign files. There can be many features in the portable executable file(PE) ‘The **Portable Executable** (PE) format is a file format for executables, object code, DLLs, FON Font files, and others utilized in 32-bit and 64-bit versions of Windows operating systems. The PE format is a data structure that epitomizes the information necessary for the Windows OS loader to manage the wrapped executable code’ [3]. Passing a PE file through sandbox or parser features can be extracted. There can be two types of malware analysis:

1. Static malware analysis
2. Dynamic malware analysis

2.1.1 Static malware analysis

If static malware analysis is done, it means the malware file isn’t running, it’s in static mode. Just parsing it through any parser will give the relevant features. This is one of the safest malware analysis techniques; one doesn’t need to create a separate isolated and safe environment, whereas in other cases a separate environment is needed because of the chances that a malware if escaped during analysis might affect the whole computer system. Using static analysis will give us features like its signature etc. Although static analysis is safe and easy to do, this type of analysis lacks to give us enough facts about malware’s behavior.

2.1.2 Dynamic malware analysis

Using dynamic analysis, we can gain information more than static analysis. In dynamic analysis malware file is executed and its behavior is analyzed. Dynamic analysis is performed in an isolated controlled environment so that there is not any chance a malware may escape from the environment created and infects the rest of the computer system. Dynamic malware analysis works better for advanced malware types.

2.2 Related work

Starting from 2017 tremendous amount of work is done on static features. Techniques applied typically involved deep learning and machine learning. One of the significant problems faced by authors was the limited amount of dataset.

2.2.1 Binary Classification: Machine learning and deep learning

In a paper “malware classification using machine learning and deep learning” dataset included 11,668 malware files collected over a span of 11 months and 2,819 benign files While feature extraction few of the malware files were corrupt therefore they were discarded and a total of 11,308 malware files were used to generate the feature vector. Features were extracted using 1 and 3 layers auto encoders independently. Opcode frequency is used as a discriminatory feature. Cross-validation was used to form a generalized model (3 fold cross-validation), to overcome the class imbalance problem. To have optimal no. of features various alternatives of dimensionality reduction were utilized (None, Variance Threshold, Auto-Encoders (1 and 3 layer auto-encoder).Models used for classification were random forest, DNN with 2,4, and 7 hidden layers. Deep learning models use ELU activation, the last layer use sigmoid activation ‘Adam’ optimizer, cross-entropy loss function, drop- out rate of 0.1,120 epochs for training. The evaluation measures used were accuracy, recall, selectivity, and precision. Random forest with variance threshold outperforms deep neural networks i.e. random forest gives 99.78% accuracy [16].

“A study on the effect of feature selection on malware analysis using machine learning” included 149 samples in total including 68 malicious files. Dynamic and static features are used. Feature selection was done using an information gain algorithm. Classifiers used were LogitBoost, Decision Table, Random Forest, LazyIBK, Multilayer Perceptron (MLP), Bayesian Classifiers, and in unsupervised techniques, Estimation-Maximization was used. The algorithms were evaluated without and with a feature selection algorithm to evaluate the performance without and with feature selection. Evaluation measures included: accuracy, confusion matrix, AUC, F1 score, training time. MLP gave the best result with an accuracy of 77.1812% [17].

The paper "malware detection approach based on artifacts in-memory image and dynamic analysis". Uses dataset comprises of 1200 PE, with 400 benign samples. Two types of features used i.e. memory analysis done using Volatility tool and dynamic Analysis done using Cuckoo Sandbox. Classifications done were done upon three types of features i.e. features from dynamic analysis features from memory analysis. Combining features of dynamic and memory analysis (Reported best results). Classification algorithms used included SVM, Naïve Bayes, KNN, random forest, decision trees. Evaluation measures included True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), detection rate, true positive rate, and accuracy. SVM gave the best results with an accuracy of 98.5% and a false positives rate of 1.7% [18].

In the paper "A novel machine learning-based malware detection and classification framework" static and dynamic features are extracted using Cuckoo Sandbox. The experiments use two types of datasets micro dataset for classification and Macro dataset for detection. Dataset is created by gaining malware benign files from Virus Total and Virus Share as a dataset in this domain is found in fewer amounts, therefore the authors propose to create a new dataset. Due to vulnerabilities that can be caused by the malware files Cuckoo sandbox is used within a virtual machine. After the features extracted by the Cuckoo Sandbox, for even better results feature selection algorithms of Chi-Square and Random Forest. Dataset comprises of 1200 samples with 522 benign samples and 678 malicious samples. 70/30 split for training and testing. Algorithms used included KNN, Random Forest Decision Tree, Random Forest, SVM. Evaluation measures included F-measure, precision-recall, AUC and accuracy. For detection and classification decision trees give the best results with 99.87% accuracy for classification and 99.37% accuracy for detection with 108 samples selected [19].

"Static and Dynamic analysis using machine learning" proposed to extract static features using PEFILE file parser provided by python. Dynamic Features are extracted using Cuckoo Sandbox.6 features were extracted using Dynamic Analysis including summary information, Registry key, IP address and DNS queries, access URL, and API calls during execution. Above 92 static features were extracted. Static features extracted from 39000 malware files and 10,000 benign files. Dynamic features extracted from 2200 malware files and 800 benign files. Non-identical dynamic feature combinations are used for training and testing purposes. Algorithms applied on both dynamic and static features included decision tree, logistic regression, bagging classifier, random forest, bagging classifier, tree classifier, AdaBoost classifier. The area under the curve for static features it gave 99.36% accuracy better than 94.64% with static features [20].

The paper named "Applying convolutional neural network for malware detection" used a five steps approach at first files are malware samples are collected and labeled. In the second step, all the file codes are converted into an image sample for applying CNN. In the third step, training takes place, in the fourth step the prediction model is built and at the final step, testing takes place as whether the test sample is either benign or malware. File package is encoded using ASCII and then converted into image samples for CNN training and testing. Inception v3 is used for the final detection of malware and benign on the test sample. Dataset comprises of 10,849 malware samples and 11,153 benign samples collected from Github, GUN open source project, honeypot, malware knowledge base, and NCHC. Evaluation measure included accuracy, True Positive Rate, and False Positive Rate. One of the research gaps was that it detects camouflaged malware embedded in benign files [21].

In the paper "Malware detection using Opcode Trigram sequence using SVM." samples are collected and the trigram sequences and the PE Headers features are extracted after the sample collection, the dataset is compiled and spitted into training and testing samples later onwards SVM is applied. Dataset comprises of 399 malware samples and 111 benign samples. Malware samples are collected from online repositories of GitHub and the benign samples are downloaded from Cnet. Opcodes are used instead of n-gram sequence as variable names and arguments change due to polymorphism and metamorphic transformation, hence opcodes are more robust to these changes. The disassembling of the file and breaking it to form trigram sequences forms the basis of this paper. They use a python script to perform this task of forming trigram sequences. The experiments use a 70/30 split for training and testing respectively. Evaluation measures include confusion matrix and statistics which include accuracy, balanced accuracy, etc. Accuracy is found to be 89.47% and a confidence of 95% 98% detection rate for malware detection [22].

The paper "feature optimization for run-time analysis of malware in windows operating system using a machine learning approach" uses dataset comprising of 236 samples having 115 benign and 121 malware samples. For feature selection, genetic algorithms are used, and furthermore 3 machine learning algorithms are used i.e. support vector machines, random forest, and support vector machines. The experiments follow a similar pipeline followed previously that is sample collection and then dataset generation using the cuckoo sandbox next feature extraction and then they perform the genetic algorithm to use only the best features for machine learning algorithms to be applied next. For Cuckoo sandbox, a virtual machine is used to provide a shielded environment. Cuckoo Sandbox uses a host and multiple guest machines. Cuckoo generates the genetic algorithm and raw features extract the significant features used for further analysis by machine learning algorithms. Also, a 10- fold cross-validation is used for better data splitting for testing and training. Evaluation Measures included confusion matrix, sensitivity, and specificity; accuracy and ROC curves. Highest accuracy of 86.6% was achieved with random [23].

"Accuracy improved malware detection method using snort sub-signatures and machine learning techniques" this paper uses multiple feature selection algorithms and works with n-gram sequences. To select the significant n-gram sequences following feature selection algorithms are used i.e., CfsSubset, Chi-Square, Principal Component Analysis (PCA), GainRatioAttribute(GR), and InfoGainAttribute(IG) .Naïve Bayes.J48, Support Vector Machines(SVM), AdaBoost IM, and Instance-Based K(IBK) is used for classification. For the training dataset, a total of 3,622 files are collected comprising of 1,971 benign files and 1,651 malware files consisting of three malware families' i.e. 550 files of Trojan family, 551 virus files, and 550 worm files. For testing, a new dataset is compiled comprising of 257 malware files and 200 benign files that sums up to be 457 files. On the basis of previous experiments done by people, the n-gram sequence of size 4 was selected as it gave the most suitable classification results. hexdump utility is used for data normalization. 10-fold cross-validation is used. Evaluation measure includes True Negative Rate, False Negative Rate True Positive Rate, and False Positive Rate. SVM has produced better results than other algorithms used [24].

"Malware Detection and Classification based on n-gram similarity sequence". This paper deals with one of different approaches rather than using only static, dynamic features or by the hybrid technique of both dynamic and static features, they propose a technique to check the similarity of n-grams sequence of malware samples with benign samples. Also they compare their results with multiple machine learning algorithms such as Bayesian Networks, C4.5 decision tree, naïve Bayes, and Support Vector Machine. The methodology follows two-steps first it withdraws the relevant features and then it selects the significant features, the features with distinguishing ability between benign and malware samples-gram sequence of training samples are gathered with n equals to 3 which means 3-gram sequences of hexadecimal file. For similarity measurement, calculate average feature vectors of training sample of training and testing samples. After taking the average, for easiness values in average feature vector the values greater than 0.5 are set to ‘1’ and below than 0.5 to ‘0’3 datasets are used for evaluation, first comprises of 88 malware and 84 benign files, second comprise of 200 malware and 168 benign samples and the last one comprises of 1000 samples each, The first dataset is used for training whereas the other two are used for testing. A total of 376,942 3-gram sequences are trained. This approach gave more convincing results than other algorithms with 95.4% True Positives and 92.61% accuracy on second dataset and 86.69% True Positives and 81.07% accuracy on the third dataset [25].

“Malware-detection model using the learning-based discovery of static features” this paper also tries to overcome the problem of signature-based malware detection. This paper works with both packed and unpacked executables. Firstly they determine whether the sample is packed or not. In the next step, data is split into testing and training data. The traditional machine learning pipeline is followed. Feature engineering is done for feature extraction from packed and non-packet executables. Feature reduction is also done to keep only the most significant and valuable features. Feature reduction is done using Principal Component Analysis (PCA), variance threshold, univariate selection, and feature importance. This paper uses ensemble learning which reduces variance and biasness. Bagging and Boosting methods are used in bagging random forest, and extra trees whereas in boosting gradient boosting and AdaBoost to reduce bias..In the detection stage 3 steps are followed i.e. noise reduction (remove unwanted data), Next packed and un-packed executables are separated furthermore in step 2 the un-packed executables are further refined for classification of whether they are malicious or not. In the last step, based on the learning algorithms of machine learning samples are defined as benign or malware.

Also, 10-fold cross-validation is used on each algorithm. The dataset comprises of 60,000 malware and 40,000 benign samples collected by the authors themselves. Evaluation measures used are confusion matrix, ROC curve, and classification accuracy. With ensemble learning results for packed executables, best results are found using univariate selection and feature importance. For non-packed executables, best results are found with the variance threshold. Accuracies obtained were above 95% for each algorithm in all the scenarios [26].

One of the papers named "Intelligent Malware Detection using Oblique Random Forest Paradigm" The proposed technique comprises of using a combined form of SVM based decision tree. It searches for the optimal hyperplane. Nodes of the decision trees are split using the hyperplane which makes it easier to split the node child. The ensemble size here acts as a hyperparameter. Results are evaluated on 3 datasets first comprising of 5210 samples with 2722 malware and 2488 benign belonging to the ClaMap dataset. The second dataset is from an antivirus dataset with 7050 benign samples and 5090 malware samples and the last dataset is from Kaggle dataset having 10866 malware samples only. A number of decision tree variants are tested which are LMT, Hoeffding Tree, random forest, J48, and oblique random forest, and REP Tree. Oblique random forest gives the best accuracy with 99.14% accuracy on the first dataset, 99.23% accuracy on second dataset and 99.52% accuracy on the third dataset. Evaluation measures used are accuracy, sensitivity, f-measure, recall, precision, and specificity. One of the contributions of this paper is the use of hyperplane to divide data into sub-nodes [27].

2.2.2 Multi-Class Classification: Machine Learning and Deep Learning

One of the papers uses birch clustering named "Malware family identification with BIRCH clustering". This paper uses clustering, most of the papers use labeled datasets and supervised algorithm techniques whereas this paper uses the BRICH clustering algorithm and tries to work with unlabeled datasets and clusters malware in their respective families. Features are extracted using Cuckoo sandbox and all non- unique feature values are removed in more than 80% of the samples. Later onwards with these samples, the BRICH clustering algorithm is used which clusters malware w.r.t their families. Dataset comprises of 5351 samples, the dataset is collected by the authors themselves. The authors also compare the performance of their results with the following other clustering algorithms DBSCAN, Hierarchical algorithms, K-Means, Mini batch K-means, Expectation-Maximization (EM) and clustering using a representative. All these algorithms work w.r.t the specific features, some algorithms performed better than others as features were up to mark for them. Evaluation measures used included the Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), Fowlkes-Mallows Index (FMI).BIRCH gave the best results for malware clustering in their respective families [28].

In the paper "Towards Efficient Malware Detection and Classification using Multilayered Random Forest Ensemble Technique" The authors claim to have produced results even better than machine learning and deep learning techniques. They used two datasets one is Malimg dataset having 25 families consisting of 9339 grayscale malware patterns. The second dataset is the dataset of Microsoft Malware Classification Challenge (BIG 2015) which contains 21,741 malware samples belonging to 9 malware families with 10,873 samples for testing and 10,868 samples used for training. The datasets used also use 1043 benign samples. The proposed method uses a layered stacked structure which is one of the ideas from deep learning. The proposed method uses 2 steps, in the first step depth-wise raw features are analyzed using sliding window of different sizes; these features are then forwarded to ensemble models. In the second step, multilayer ensemble methods are then used in the second step, algorithms used for ensemble modeling are Random Forest, Extra Tree Classifier, XgBoost and Logistic Regression. In the stacking phase, the images are divided into two sizes, 10 x 10 and 30 x 30. These images are then used by ensemble methods. The probability feature vectors calculated at each layer are forwarded to the next ensemble stacked layer. In multilayered each layer contains all 4 ensemble algorithms. Results are combined with the first layer results and fed to the next layer. This goes on until the results of the layers start getting the same. In the end, the highest probability is checked. Evaluation measures used are accuracy, confusion matrix, f1-measure, precision, and recall. In all the scenarios accuracy achieved is above 95% [29].

The paper named "Malware Classification using machine learning algorithms" uses the following algorithms multi SVM, multilayer perceptron, and decision tree,. The dataset contains features of different types of malware. No. of samples used is 5724.

The dataset contains 7 types of attributes. For categorical values, class names are represented as numbers. There are 10 classes; each class is a malware of different types. Feature importance and confusion matrix is used. SVM gave an accuracy of 90.2% and the neural network gave an accuracy of 98.94% [30].

In a paper similar to the paper above named "Malware classification using machine learning algorithms and tools", this paper tried to overcome the problem of polymorphic nature of the new malware that are being created. Dataset used is from 5 different families of malware (Zeus, APT1, crypto, Locker, and shadowbrokers), after all the data collection and pre-processing the number of remaining malware samples counted to be is of 5800 samples.6 machine learning algorithms are used that are decision tree, naïve Bayes, KNN, random forest, neural network, and SVM.

Also, 4 tools are used for more analysis tools used are Weka, RapidMiner, Knime, and Orange. Evaluation measures used are accuracy, confusion matrix, and Cohen's Kappa.In terms of algorithms random forest gave the best results and in terms of tools used orange gave a better result than knime with an accuracy of 94.2% and Cohen's kappa equal to 93.8% using orange tool and random forest algorithm [31].

Using convolutional gated neural network the authors use the dataset of Microsoft malware classification challenge provided in 2015(Big 2015). Dataset comprises of 21,741 samples with 10,873 samples for testing and 10,868 samples for training. The dataset has 9 classes of malwares and no benign samples. Dataset has .asm and .byte file for every sample. Only binary data is used in this paper. One hot vector is used for each possible x86 instruction. The proposed technique uses, DNN (Deep Neural Network), CNN (Convolutional Neural Network), and GRU (Gated Recurrent Network). CNN uses convolutional layer, activation layer, and then the pooling layer. CNN layers output a single layer which acts as input to the GRU layer. GRU outputs as per no. of inputs received. GRU output serves as input to DNN. DNN layer outputs vectors of values. Finally, the sigmoid layer is placed which classifies malware into one of the 9 malware families. The proposed model gives 92.66% accuracy. The paper is named as "Classifying malware using convolutional gated neural network" [32].

Using deep learning methods a paper named "Malware classification using deep learning methods" this paper uses a shallow deep learning method based upon word2vec space model representation for malware representation and later onwards gradient boosting machine (GBM) is used for classification. The dataset used is taken from Kaggle, the dataset is provided by Microsoft Named Microsoft Malware Classification Challenge (Big2015). To test the model performance in another way 100,200,300 and 398 files from each class are taken except 5th class as it doesn't have enough samples. This step is taken to prove the performance of the model on a small amount of data. Assembly codes are stripped from their arguments so that only assembly operation sequence can be acquired separately. Evaluation is done using logloss values. The highest error rate was reported by 100 samples and a minimum error by 398 samples. GBM is performed upon these 4 sets of samples (100,200,300,398). 5-fold cross-validation is also performed [33].

In the paper "Analysis of ResNet and GoogleNet models for malware detection" dataset used is from Microsoft Malware Classification Challenge (BIG-2015). Also, 3000 clean benign software are downloaded from open source websites. Byte files are converted into their equivalent images. Poly Unpack(used for dynamic code), PEID (used for static code).The opcode is de-compiled from the assembly codes and is grouped in 2-tuple opcode grouping. Binary images are reconstructed from the opcode sequence with the help of information gains and their probabilities. Probabilities and information gains are calculated based upon the frequencies of opcodes. Images are constructed based upon binary opcodes frequency. Normalization, dilation, and erosion of histograms are used for better enhancement of opcode sequences. Normalization is done through by dividing with the maximum value. The generated dataset is split into testing and training, after then GoogleNet and ResNet are utilized in comparison. ResNet gave much better results than GoogleNet. ResNet-125gave an accuracy of 88.36%, training accuracy of 87.98%, and 11.94 loss[34].

2.2.3 Image Based Malware Classification

This paper named "Malware Detection with Malware Images using Deep Learning Techniques" uses the basic pipeline of data preprocessing, classification, and evaluation. Dataset is provided by Korea university from the Andro-dumpsys study. The dataset comprises of 906 malware samples which comprise of 13 different malware families and 1776 benign files downloaded from Google store and parsed through Virus Total to check its benign nature. Data is pre-processed by converting byte values into an equivalent grayscale image and also they have converted into RGB images. For this, a batch of 3 bytes will represent one pixel. An image of 1920 pixels is generated. Before classification, the problem of the image downsampling occurs. To overcome this SSP (Spatial Pyramid Pooling) is used which takes into account. For classification two types of models are proposed one is a simple 3-layered CNN model and the other one is ResNet-50.Before final layer batch normalization and dropout is used for both models. For evaluation, a threshold value is taken above which are malicious, and below are benign. Also for evaluation, an image is divided into subparts and if anyone of those sub-images is classified as malicious, that complete image is considered as malware (this approach is taken to .use less memory). Models were also tested with unaltered samples and adversary samples. In results ResNet-50 performs better with RGB and 3-layered CNN performs better with grayscale images. Added API codes don't work well with RGB as it shifts the channels [35].

A deep convolutional neural network is proposed by the authors of the paper named "Malware Classification using Deep Convolutional Neural Networks" for malware classification. Malware files are converted into their equivalent grayscale images by converting the binaries into an 8-bit vector and then converting those 8-bit vectors are converted into decimal values. The resulting matrix is shaped into a 2d- grayscale image. The CNN model used is based upon the architecture of VGG-16 and is called M-CNN.6 blocks are used with different arrangements and no. of layers in it. Layers include Convolutional Layer, Relu that is the activation layer, and max pooling. After that flatten layer is used 3 times and for final classification, softmax is used. Datasets used are the dataset of the Malimg dataset contains samples for 25 families having 9,339 malware samples in total. 10% are used for testing and 90% for training. The other dataset used is the dataset provided by Microsoft name Microsoft Malware Classification Challenge. This dataset comprises of 21,741 samples in total with 10,868 for training and 10,873 for testing. For each sample, this dataset provides a .byte file and a .asm file. Only byte files are used to generate images. Classification for Microsoft Malware Classification Challenge is done upon two sets. 1st sets use 90/10 the ratio for training testing and the other set contains the original distribution of training testing given by Microsoft. With both datasets accuracy stated is above 98% [36].

A deep convolutional neural network is used in the paper" Malware Classification using Deep Convolutional Neural Networks". Dataset used is the dataset provided by

Microsoft. Dataset is further passed through IDA which returns assembly code and byte data. Next is the image generation portion, instead of using assembly code byte data is used for image conversion. The hexadecimal values in byte data are converted into a string of binary values through which hamming distance will be calculated. As a result these binary string values are merged into equivalent binary images. Hamming distance is used to check the average string length. Here hamming distance is used to determine the optimal length for image breakdown for further processing. Autocorrelation is calculated based upon the hamming distance between bytes for 600 instructions. This paper uses VGG16, VGG19, and AlexNet, and they accept RGB image input for that the generated greyscale images are replicated across the RGB axis to serve the purpose of input. After image creation features are extracted and features are inputted to AlexNet, VGG 16, VGG 19 heterogeneous and also inputted separately to 3 AlexNet homogeneous. Side by Side transfer learning is also used giving its output to a single network and after training of this single network final classification layer is removed and new layers are added for new dataset. For final classification from both the parallel models SVM is used. In the results transfer learning method outperforms the other methods [37].

Xception based convolutional network is used to overcome the problem of overfitting. Dataset used is the dataset of the Malimg dataset. Another dataset is used supplied by Microsoft. As Microsoft Malware Classification Challenge uses .asm and .byte files for each sample binary stream is read from byte files and for .asm files they are read as binary streams. A binary stream can easily be segmented into 8 bit vector; further 8-bit vector can represent a greyscale pixel value. Transfer learning is used with Xception Image Net. Also, an ensemble model is proposed. The ensemble model is only 3- layered model.Xception-1 was trained with byte files only and Xception-2 was trained with .asm files only. An ensemble model was used.The evaluation measures used were logloss and accuracy. Results produced were either equivalent to or better than the results mentioned in the literature, but training time was reduced to approximately 5 minutes only [38].

A ResNet-50 based deep neural network is proposed in the paper "Malicious Software Classification using Transfer Learning of ResNet-50 Deep Neural Network". The data set is Malimg. Each malware executable is converted into its equivalent byteplot greyscale image. Furthermore, these grayscale images are rescaled to 224 x

224 for greyscale to RGB conversion. Images are generated on the base of an 8-bit vector. The proposed model is based upon ResNet -50, the last 1000 layers are replaced with 25 softmax layers. Pre-trained weights of ResNet-50 trained on Image Net dataset are used In results an average accuracy of 98% was achieved. Also, better accuracy was achieved with those that were stated in the literature [39].

In the paper "Deep Learning Framework and Visualization for Malware Classification", a hybrid network is used, combining CNN and LSTM. Dataset used is the dataset of Malimg. To use this dataset malware samples were transformed into equivalent grayscale images. Malware binaries are converted into their equivalent 8-bit vectors and those 8-bit vectors are then converted into their equivalent 2D gray scale images. A 70/30 split is used for training and testing. In the model, CNN is followed by LSTM layer which extracts the features. Mask size for convolution was taken to be 3 after experiments. 2 layers of convolution use 64 and 128 filters each which are inputted to LSTM. After LSTM a fully connected layer is utilized for classification. Evaluation measures used were accuracy, precision, recall, f1-score. Accuracy was achieved to be 95%. The paper uses results from another paper in the literature that uses CNN with SVM which has reported 84% accuracy, whereas this model gave 95% accuracy [40].

In the paper "Malware detection using malware image and deep learning" the basic pipeline of training testing is used. Malware and benign samples are converted into their equivalent grayscale images and are trained and tested using deep learning models. For image conversion, every 8 bit is converted into a pixel value and results in a 2D grayscale image. An image of size 256 x 256 is used which turns needed to be downsampled to 32 x 32 size for further processing. A simple convolutional neural network is proposed.3 convolutional layers, pooling layers are used and finally, 2 fully connected layers are used for final classification.10,000 normal files were taken from Hauri [41] and 2000 malware files from Kaist Cyber Security Research Center [42]. These files are converted into equivalent greyscale images using the technique described above. A 90/10 split was used for training and testing. In results, 95.6% accuracy was achieved [43].

Dataset used in the paper "Malicious Classification Based on Deep Learning and Visualization" is the dataset of Microsoft Malware Classification Challenge. The dataset contains .asm and .byte files for each sample. byte files are converted into equivalent greyscale images using the 8-bit vector technique. In the model 5 convolutional layers are used and after that Spatial Pyramid Pooling is used before the final fully-connected classification layer. 10 folds cross-validation is used. The final average accuracy was reported to be 95.44% [44].

Summary

Related work is discussed in the chapter, it can be seen that using static features quite the amount of work is done. Many researchers use the combination of dynamic and static analysis or dynamic analysis with memory analysis. Research done with machine learning, the results produced were quite reliable, but cross-validated models were not seen in the literature generally single dataset is used for training and testing.

CHAPTER 3

DATASET ACQUISITION

This chapter deals with the procedure followed for dataset collection for this thesis work. Two sets of datasets are used in this thesis work.

3.1 Publicly available datasets

There are many different datasets available for malware analysis. Datasets use different types for feature including static feature, dynamic features, a hybrid approach of both static and dynamic features, API calls, datasets based upon memory analysis etc. For this thesis dataset used is based upon static features.

On the pattern of static features some of the datasets publically available are one of the dataset includes 73,775 binaries dedicated solely for malware analysis based upon static features only.

To work with models provided by deep learning, a decent amount of dataset is used. Most authors use dataset of 10,000 samples or above for deep learning classification models.

The dataset used in this thesis work is publically available dataset provided by “Chiheb Chebbi”.Chiheb Chebbi is an author of two books named “Mastering Machine Learning for Penetration Testing” [4] and “Advanced Infrastructure Penetration Testing” [5]. Chiheb Chebbi is an InfoSec enthusiast having expertise in many cyber security-related researches. His core interests lie in deep learning, penetration testing and malware analysis. He is a well know researches his proposal talks were accepted by BlackHat Europe 2016, DeepSec 2017 and many other well know information security conferences [6].

Dataset provided by Chiheb Chebbi is used in one of his books and is available to everyone for use. Dataset comprises of a total 138,048 binaries having 96,724 malware binaries and 41,324 benign binaries. The datasets comprises of 36 static features only.

The book authored by Chiheb Chebbi “Machine Learning for Penetration testing”, using this dataset Chiheb Chebbi has applied few machine learning algorithms to make the readers familiar with the concept of malware classification using machine learning.

This dataset became public in the year 2018; so far this dataset is used only in the book published by Chiheb Chebbi.

3.2 Self-Collected dataset

The dataset collection phase was a difficult task. Collecting malware portable executable files was a very risky task also collection of benign portable executable files was a length process.

For malware portable executable files collection different approaches were used. First one was the use of low-interaction and high-interaction honeypots. Low and high interaction honeypots were used to gain access to maximum amount of malware portable files. Other than that the malware executable files that were being stored by Kaspersky anti-virus database were also utilized. After this the files were also cross-checked with VirusTotal [7] to insure that they were actually malware files. On the other hand the benign files were collected from different machines with licensed and updated versions of windows operating systems including windows 7,8 and 10. Also the benign files were also cross checked for their labels.

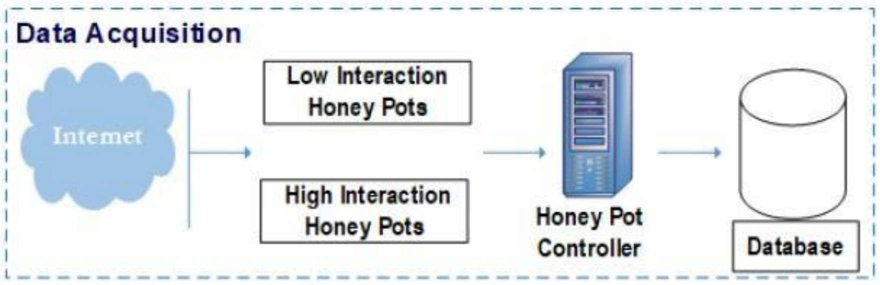


Fig 3. 1: Data Acqusition

The next step was to extract the respective features. Only static features were extracted but, due to the malicious nature of malware files a separate virtual machine was used. The virtual machine workstation used was VMware workstation 15.0 Pro [8], the virtual machine used within this workstation was Linux-Ubunto. 64 - bit Ubunto iso image [9] was used for installation on a virtual workstation. Furthermore, Jupyter Notebook [10] was used for static analysis. Using the library 'pefile' [11] provided by python static features were extracted and stored in an excel file. Similarly, static features were extracted from benign files and stored in an excel file.

With data collection, two types of redundancy was seen. While downloading the samples redundancy was seen in the names of the files, a lot of repetition was seen in the files based upon files with similar names. Secondly, after extraction of malware and benign features in a single excel file huge amount of redundancy was seen in the feature named 'md5'.Md5 is a feature that stores unique signature value of the file, hence it means there was a lot of redundancy in the files, after using a filter upon the feature of md5 a total of 21,486 samples were left with unique md5 values. This dataset includes 14,497 malware samples and 6,989 benign samples. In the figure below is shown the flow of the dataset collection phase.

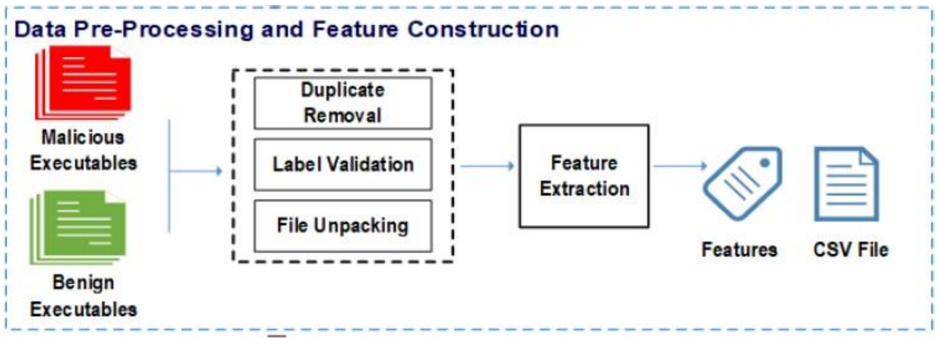


Fig 3. 2 : Data cleaning pipeline

3.3 Features used in dataset

For this thesis work, only static features are used. The following features are extracted from Portable Executables (PE) and used in the dataset used for thesis:

1. MD5
2. Machine
3. Size of optional header
4. Characteristics
5. Major Linker Version
6. Minor Linker Version
7. Size of Code
8. Size of Initialized data
9. Size of Uninitialized data
10. Address of Entry point
11. Base of Code

12. Image Base
13. Section Alignment
14. File Alignment
15. Major Operation System Version
16. Minor Operating System Version
17. Major Image Version
18. Minor Image Version
19. Major Sub System Version
20. Minor Sub System Version
21. Size of Image
22. Size of Headers
23. CheckSum
24. Sub System
25. Dllcharateristics
26. Size of Stack Reserve
27. Size of Stack Commit
28. Size of Heap Reserve
29. Size of Heap Commit
30. Loader Flags
31. Number of RVA and sizes
32. SectionsNb
33. Load Configuration Size
34. Debug Size
35. Resource Size
36. Export Size
37. Section Minimum Entropy
38. Section Maximum Entropy
39. Section Mean Entropy

Below are discussed the details of these features

3.3.1 Md5

Md5 is a cryptographic signature. Md5 is a 32 bit hexadecimal value and each file has its unique Md5 value. Md5 value is generated when the file is created.

3.3.2 Machine

This number represents the target machine.

Table 3. 1: Static feature named “machine” values [12].

Values	Decimal Values	Description
0 x 0	0	It is assumed that contents are applicable to any machine type

Values	Decimal Values	Description
0 x 14c	332	Intel 386 or later and compatible processors.
0 x 166	358	MIPS® little endian.
0x184	388	Alpha AXP™
0 x 268	616	Motorola 68000 series
0 x 1F0	496	Power PC, little endian.
0 x 290	656	Precision Architecture (PA) RISC processor.

3.3.3 Size of optional header

Irrespective of the fact that the feature name says it to be ‘optional’ whereas in reality this is a mandatory feature which gives information related to portable executable file.Size of the optional header, is included only for executable files and not object files. Value of 0 should be here for object file. Size of optional header is for 32 bit files is 224 bytes. It has a size of 240 bytes, for 64 bit files [12][13].

3.3.4 Characteristics

These flags indicate attributes of the file. The table below shows the hexvalues, their equivalent decimal value and what information it holds about the attributes of the file.

Table 3. 2: Characteristics values [13]

Values	Decimal Values	Description
0 x 0001	1	This flag value indicates that file hasn’t got base relocation therefore it should be loaded at its preferred base addressed. The loader reports an error if base address is unavailable..

Values	Decimal Values	Description
0 x 0002	2	Linker normally indicates an error if this flag isn't set. Generally this flag means that the image file is well-founded and can be run.
0 x 0004	4	COFF line numbers have been detached.
0 x0008	8	Local symbols have been detached for COFF symbol table entries.
0 x 0010	16	Aggresively trim working set.
0 x 0020	32	Addresses greater than 2 gb can be handled by app
0 x0040	64	Reserved for future use.
0 x 0080	128	Little endian: LSB precedes MSB in memory.
0 x 0100	256	Machine based on 32-bit-word architecture.
0 x 0200	512	From image file debugging information is detached.
0 x 0400	1024	If image is on removable media, copy and run from swap file.

Values	Decimal Values	Description
0 x 1000	4096	Image file is a system file, not a user program
0 x 2000	8192	The image file is a dynamic-link library (DLL). These types of files are considered executable for all types of purposes, but cannot be directly executed.
0 x 4000	16384	Only on a UP machine this file should be executed.
0 x 8000	32768	Big endian: MSB precedes LSB in memory.

3.3.5 Major/Minor Linker Version

The VERSION option tells the linker to place a version number in the header of the .dll or .exe file. The values of major linker version will be significantly different from each other in malware and benign files [14].

3.3.6 Code Size

Size of the code (text) section, if there are multiple sections then sum of all the code sections.

3.3.7 Size of Initialized data

Size of the initialized data section, in case there are multiple sections, then it is the sum of all those. Size of initialized data is noticeably larger in benign files as compared to malware files [15].

3.3.8 Size of Uninitialized data

Size of the uninitialized data section (BSS), in case of BSS sections then sum of all these sections. The size of the un-initialized data is 0, generally in most of the cases [12][13].

3.3.9 Address of Entry Point

Entry point address is the address where the PE loader will begin execution; this address is relative to image base when in memory the executable is loaded. This is the starting address, for the image program; this is the address of the initialization function, for device drivers and, this is optional for the DLL [16].

3.3.10 Base of Code

This is the pointer to the beginning of the code section, which is relative to the image base [16].

3.3.11 Image Base

This is the preferred address of the first byte of the executable when it is loaded in memory. This value is a multiple of 64K bytes. The default value for applications is 0x00400000(4194304), except on Windows CE where it is 0x00010000. The default value for DLLs is 0x10000000(268435456). [16].

3.3.12 Section Alignment

When the executable is assigned some address the sections in the executable also need to be loaded. The section alignment is set to 0x2000. This means that the code section starts at 0x2000 and the section after that starts at 0x4000 [13][16].

3.3.13 File Alignment

Just like the section alignment the data also needs to be loaded present in the executable files.The file alignment indicates to the beginning of every section in the file. It is set to 512 bytes or 0x200. Therefore, the initial section starts 512 bytes from the start; the next section starts at 1024 bytes, and so on. The value should between 512 and 64K inclusive, also be a power of [12][13].

3.3.14 Major/Minor Operating System Version

The major operating system is the version supported by Portable Executable [16].

3.3.15 Minor Operating System Version

The minor operating system is the version supported by Portable Executable [16].

3.3.16 Major Image Version

Major image version is the major version number of the image [16].

3.3.17 Minor Image Version

Minor image version is the minor version number of the image [16].

3.3.18 Major Sub System Version

The major version number of sub system [16].

3.3.19 Minor Sub System Version

The minor version number of sub system [16].

3.3.20 Size of Image

Size of image is the size of executable after being loaded into memory.Size of image must be a multiple of Section Alignment, it is in bytes, of image, including all headers [12][13].

3.3.21 Size of headers

The size of the headers represents the size of all the headers, i.e. PE header, the optional header, DOS header, also, some other sections. The value is a multiple of the file alignment, i.e. 512 bytes.

3.3.22 Checksum

Checksum of the image file. These files are validated at load time: any DLL loaded into a critical system process all drivers, and any DLL loaded at boot time, [16]. The checksum is a notion that is used to confirm whether a file is undamaged or has been corrupted. Normally it is never used and value is zero , even when PE file has field for it [13].

3.3.23 Sub System

This field points to the user interface type needed from Windows; the DLL flags field has a zero value because file is not a DLL file.

Table 3. 3: Sub System Values [16]

Value	**Description**
0	Unknown subsystem
1	Subsystem is not required (native system processes and device drivers).
2	Windows graphical user interface (GUI) subsystem.
3	Windows character-mode user interface (CUI) subsystem.
5	OS/2 CUI subsystem.
7	POSIX CUI subsystem.
9	Windows CE system.
10	Extensible Firmware Interface (EFI) application.
11	EFI driver with boot services.
12	EFI driver with run-time services.
13	EFI ROM image.
14	Xbox system.
16	Boot application.

3.3.24 DllCharacteristics

Usually set to zero for malicious files [15]

Table 3. 4: DllCharacteristics values [16]

Value	Decimal	Description
0 x 0001	1	Reserved
0 x 0002	2	Reserved
0 x 0004	4	Reserved
0 x 0008	8	Reserved
0 x 0040	64	The DLL can be relocated at load time.
0 x 0080	128	. If this flag is set by you and a section contains only uninitialized data, set the PointerToRawData member of IMAGE_SECTION_HEADER for that section to zero; otherwise, the image will fail to load because the digital signature cannot be verified.
0 x 0100	256	The image is well suited with data execution prevention (DEP).
0 x 0200	512	The image shouldn’t be isolated but it is isolation aware
0 x 0400	1042	Handlers can’t be called in this image. The image does not use structured exception

		handling (SEH).
0 x 0800	2048	Don’t bind the image.
0 x 1000	4096	Reserved
0 x 2000	8192	A WDM driver.
0 x 4000	16384	Reserved
0 x 8000	32768	The image is terminal server aware.

3.3.25 Size of Stack Reserve

Sizes of stack reserve are number of bytes to reserve for the stack. Just the memory determined by the SizeOfStackCommit part is submitted at load time; the rest is made accessible one page in turn until this reserve size is reached [16]. The Stack reserve size field decides the stack region that the thread can utilize. Typically, the size is 1 MB [13].

3.3.26 Size of Stack Commit

The Stack Commit is the amount of memory that the stack is relegated at startup [13].

3.3.27 Size of Heap Reserve

Size of local heap load space to reserve. Just the Heap Commit Size is committed; the rest is made accessible one page in turn, until reserve size is reached [12].

3.3.28 Size of Heap Commit

Size of local heap space to commit [12].

3.3.29 Loader Flags

This flag informs upon loading whether to break upon loading, debug on loading or to set to default [17].

3.3.30 Number of RVA and sizes

The number of relative virtual addresses in the rest of the optional header. Each entry describes a location and size [16]. The structures contain critical information about specific regions of the PE file. The 8-bytes in the structure involve two ints, one known as the Relative Virtual Address (RVA) and the second as size[13].

3.3.31 Load Configuration size

This configuration is usually used for exceptions. The Load Configuration is only utilized in Windows NT, Windows 2000 and Windows XP [13].

3.3.32 Section Minimum/Maximum/Mean Entropy

Entropy value of specific file is represented by using the digital values of 0 to 8. Entropy is used to check whether a file is packed or not. The outcome is either near to 0 or 8 as well as in-between these two numeric values. An entropy value greater than 5 usually means that the file is malicious [11].

3.4 A Comparison of benign and malware threshold values

After the feature details of static features discussed above, it is also important to know the threshold values, i.e. for which value a file is a benign file or a malware file. Below in the table 2.5 these threshold values are given

Table 3. 5: Portable Executable discriminative features

Features	Benign	Malware
Data raw size	High	Low
Debug size	Has value	Usually 0
Check sum	Has value	Mostly 0
Number of symbols	384.2	0.002
Major image version	Has some value	Usually 0
Major linker version	8.334	4.419

Summary

In this chapter, the availability of publicly available datasets is discussed as well as the pipeline to gain access to portable executable malware and the benign file is discussed. Furthermore, the process of static feature extraction is explained. In the end, the details of the features extracted are discussed.

CHAPTER 4

METHODOLOGY

This chapter deals with the methodology used for machine learning algorithms on both the datasets described in the previous chapter. Data used is directly exported from CSV file created earlier.

4.1 Basic Flow: Methodology Part-1

After downloading the dataset and pre-processing two types of pipelines are followed. In phase-1 data is directly imported from the CSV file created previously and after normalization machine learning algorithms are applied. In machine learning phase for training and testing datasets are used interchcangeably as the model is cross-validated. Both datasets are used for training and testing vice-versa for cross-validation.

A second phase is adopted which deals with clustering. Using the same dataset from CSV file feature dimensionality is done and these features are then fed to clustering algorithms, after that a comparison is done between results deduced from different clustering algorithms. 3 different clustering models are proposed using different dimensionality reduction techniques.

In both phases normalized dataset is used. Normalization is done through min-max scaling

The first half of chapter deals with machine learning algorithms named as methodology phase-1 and the remaining half chapter deals with different dimensionality reduction techniques used for clustering named as methodology phase-2

On the next page is shown the proposed pipeline for this thesis work.

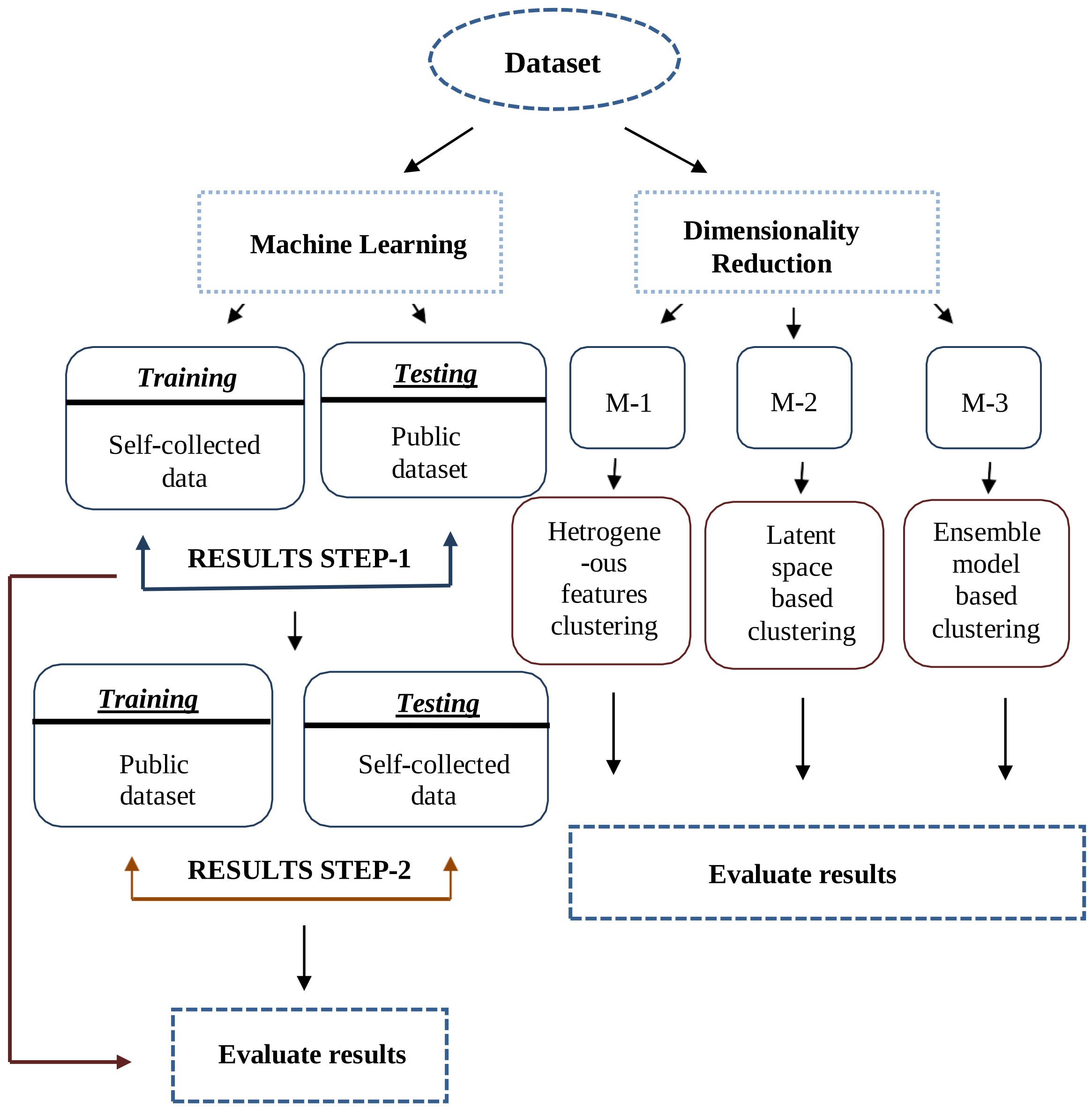


Fig 4. 1: Methodology pipeline

4.2 Goal: A robust model

The goal of using machine learning on two different datasets is to generate a robust model. Model is trained self-collected dataset, testing is done on a different dataset both uses similar static features which make this model more robust.

4.3 Strategy for Supervised Algorithms

Moving towards the pipeline followed for machine learning algorithms, the initial step is to load dataset, after that the dataset is normalized because some features have binary values; some features have continuous values whereas some features have discrete values. To overcome this problem, all data is normalized in between 0-1 range using Min-Max scaling. Below is shown the flow for phase-1.

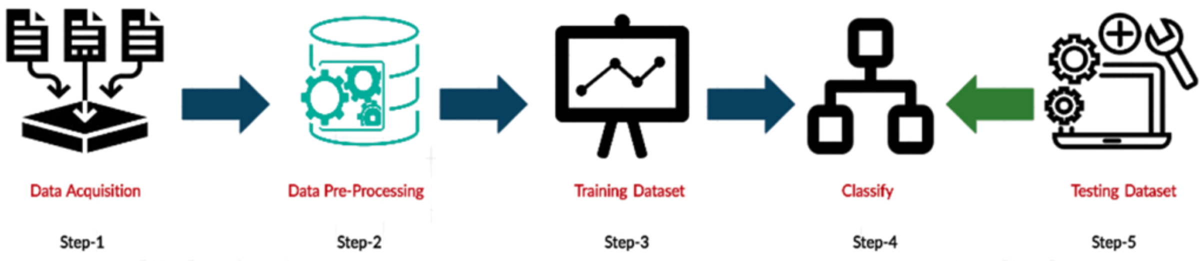


Fig 4. 2: Flow for Supervised Machine Learning Algorithms

The normalized dataset is then fed to following six machine learning algorithms for training

1- Random forest

2- Support Vector Machine (SVM)

3- K- Nearest Neighbors (KNN)

4- Gradient Boosting

5- AdaBoost

6- Naïve Bayes

After training, the test dataset is normalized on the same pattern, after that this dataset is tested on the above given algorithms. Following evaluation measures are used for evaluation

1- Accuracy

2- Confusion matrix

3- Classification report

4- ROC for comparison of all algorithms

Both datasets are used vice versa for training and testing. In step one training is done upon self-collected dataset and testing is done upon publicly available dataset in the second step datasets are interchanged for training and testing.

4.4 Strategy for Un-Supervised Algorithms

Just like unsupervised algorithms, dataset is loaded for unsupervised algorithms. In this strategy two things are done, one clustering is performed using all the features present in the dataset on the other hand clustering is done after passing it through Principal Component Analysis(PCA) feature reduction algorithm. Models are trained upon publicly available dataset and tested upon self-collected dataset.

Similarly dataset is first normalized and in reformed in the range of 0-1 to be used for clustering algorithms. Also, the feature named “md5” is a 16-bit hexadecimal number which is first converted into its equivalent decimal value and then is brought into the range of 0-1 like all other features. Similar action was taken for supervised algorithms as well.

The following algorithms were used for clustering with all features and after feature reduction done using PCA

1- K-Means

2- K- Means mini

3- Gaussian Mixture Model (GMM)

4- Fuzzy c-means

Below are listed the evaluation measures used

1- Accuracy

2- Area Under Curve (AUC)

3- F1- Score

4- Precision

5- Recall

Training is done upon publicly available dataset and testing is done upon self-collected dataset.

Below is shown the pipeline followed for un-supervised machine learning classification models.

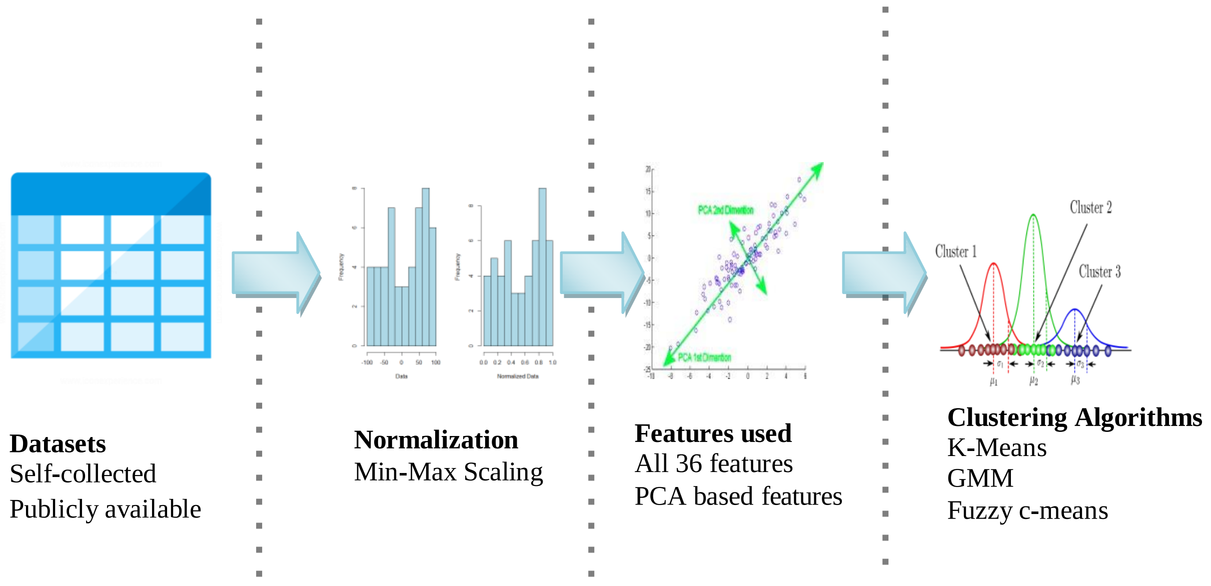


Fig 4. 3: Pipeline for Un-Supervised Machine Learning Algorithms

4.5 Basic Flow: Methodology Part-2

In this part, the dataset is used in two different ways. First, the dataset stored in CSV file is used and clustering is performed with and without dimensionality reduction. Three experiments are done, clustering without dimensionality reduction, clustering with dimensionality reduction using Principal Component Analysis(PCA), and dimensionality reduction using autoencoder. Second instead of using the extracted static features, the downloaded portable executable malware and benign files are converted into their equivalent grayscale images to be fed to deep learning classification models.

4.6 Clustering using Static Features

The self-collected static features dataset is used in this part. Using the approach of dimensionality reduction comparison is done. Below are the three experiments done and later used for comparisons

1. K-Means clustering with all 36 features, PCA and auto encoder

2. K-Means, Gaussian Mixture Model and fuzzy c-means with all 36 features, PCA and

auto encoder(using only encoder part)

3. Same experiment as above with ensemble NN blocks in auto encoder.

The following clustering algorithms were used

1. K-Means
2. Gaussian Mixture Model
3. Fuzzy c-means

4.6.1. Simple Clustering

In this experiment, clustering is done upon k-means only. The dataset is loaded and is normalized in the same manner as before. The first result is produced by clustering using all thirty-six features. Second the dataset is passed through Principal Component Analysis (PCA) and a resultant of three features is used only. The last experiment of this set was to clustering using output from autoencoder. Simple auto encoder architecture was proposed. In figure 4.4 is shown the overview of this step

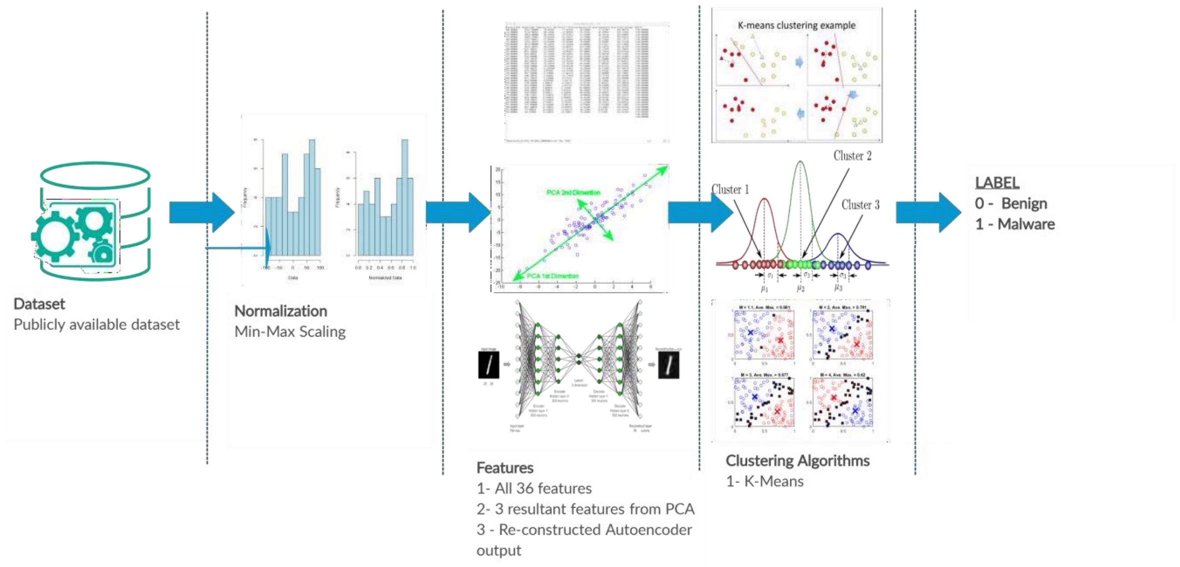


Fig 4. 4: K-Means clustering with all feature, PCA and autoencoder

In figure 4.5 is shown the architecture of the autoencoder used

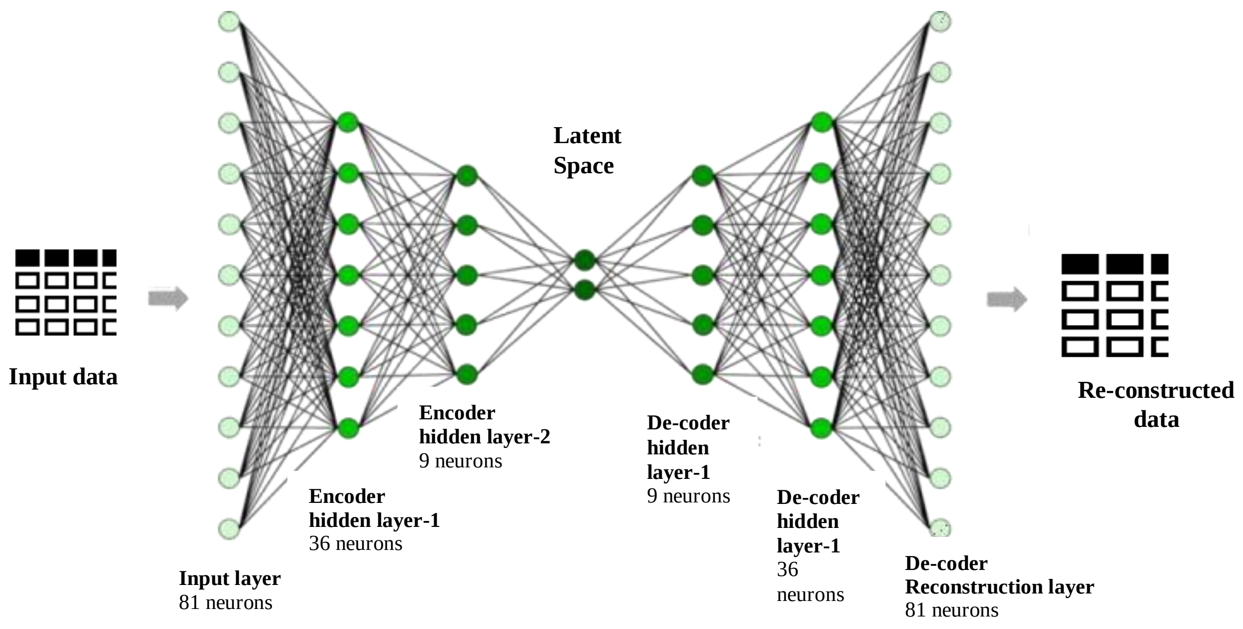


Fig 4. 5: Autoencoder architeture for K-Means clustering

4.6.2. Clustering with Autoencoder

In this experiment, the same autoencoder which was used in the previous step was used, with only one difference that decoder portion was discarded and only encoder portion was used in order to obtain results from the summarized features from all the thirty-six features. This model highly highlights the importance of latent space. Therefor latent space is used directly for clustering. Instead of using only the k-means clustering Gaussian Mixture Model (GMM) and fuzzy c-means are also used for clustering. Evaluation measures used were:

1) Accuracy
2) Precision
3) Recall
4) F1- Score
5) Rand Index
6) Jaccard Score

Similar approach was used in a paper where they have used only k-means and used only the encoder portion. Results of the paper "Static malware clustering using enhanced deep embedding method"[2] were crossed using the model described in this section. In the figure 4.6 below is shown the architecture followed for this step

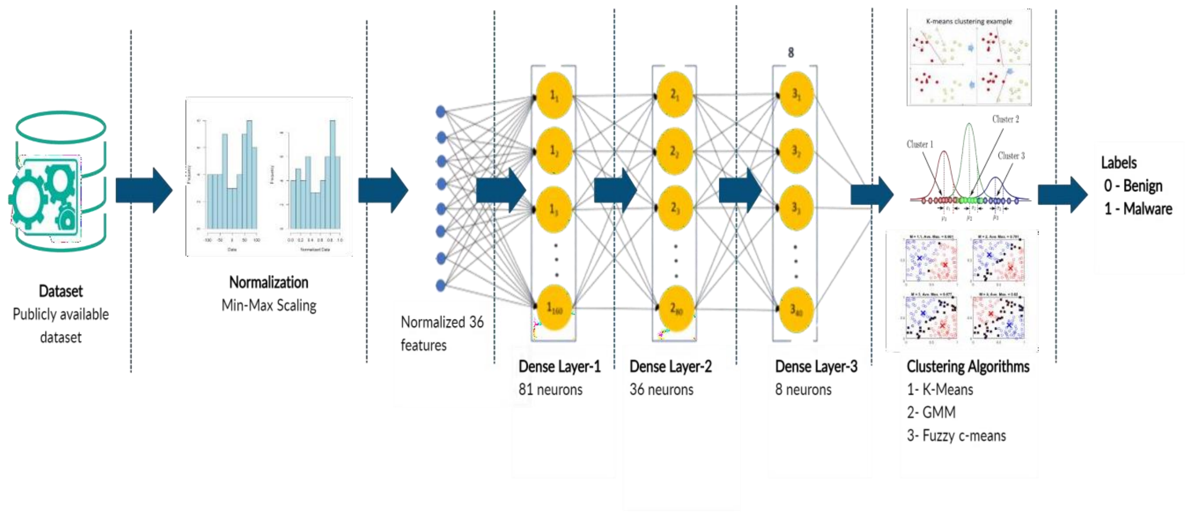


Fig 4. 6: Clustering results with encoder only

Publicly available dataset is used.

4.6.3. Clustering with Ensemble Neural Network Based Autoencoder

The final feature selection technique used is the technique of autoencoder with bagging based ensemble block. Dataset is loaded and is normalized all thirty-six features are fed to the encoder, after that the last layer of encoder is fed to three different neural networks, each of that the layer contains 4 nodes, the results from these three neural networks are averaged to be fed to final output layer. Bagging ensemble model is used instead of sequential execution bagging approach that means parallel execution is used for even more efficient classification results. The purpose of doing this was to average the most important feature in order to make the model more efficient for final classification. Publicly available dataset is used. Also, ensemble based model is used to refine the stored features in the latent space even more than as it was refined using encoder only. The parameters used for this model are shared in the next chapter. Evaluation measures used are similar to that of model-2

Below is shown the final autoencoder architecture used in this experiment

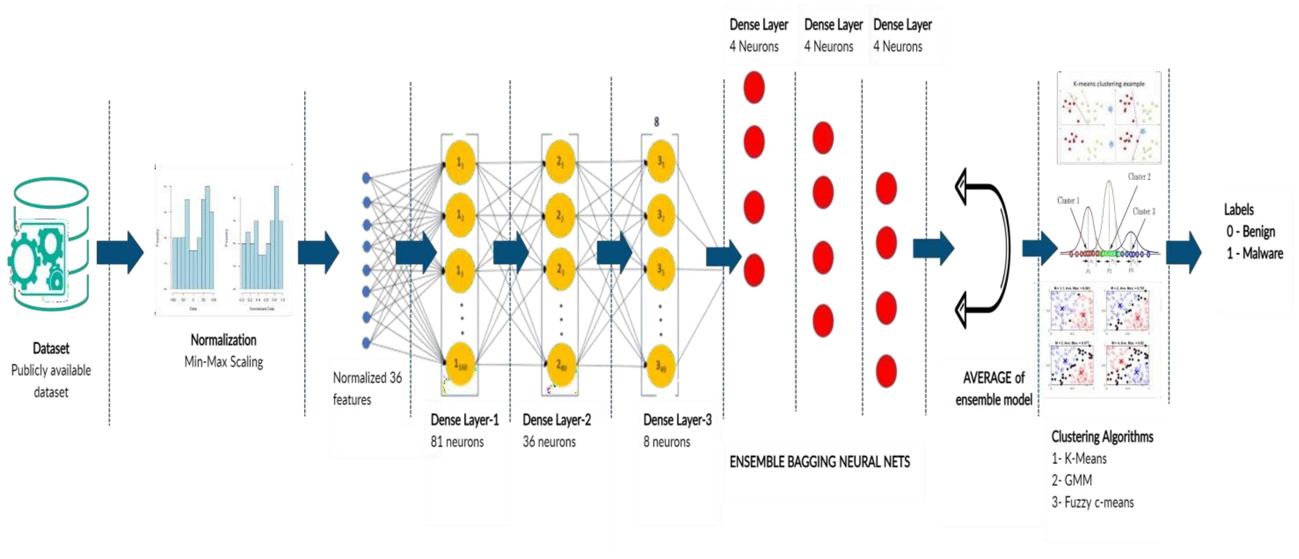


Fig 4. 7: Pipeline for ensemble Neural Network based autoencoder

Summary

Methodology is divided into two parts first part deals with classification using machine learning algorithms based upon static features, where as in the second portion of methodology clustering is done after dimensionality reduction using PCA and autoencoder. Algorithms used after dimensionality reduction were K-means, GMM and fuzzy c-means.

CHAPTER 5

RESULTS AND DISCUSSION

5.1. Parameters for Supervised Algorithms

Parameter tuning is an important step for obtaining accurate and best results. Parameter tuning is a technique in which different parameter values are used, the values which give best results are kept. Sometimes setting the values to default also gives convincible results. In the table 5.1 are shown the tuned parameter values used for supervised machine learning algorithms

Table 5. 1: Supervised algorithms tuned parameters values

Algorithms	**Parameters values after tuning**

Random forest	n_estimators=500
	max_depth=10
	random_state=0

SVM	gamma='auto'
	probability=True

KNN	No. of neighbors: 200
Gradient Boosting	Default values

AdaBoost	n_estimators=500
	random_state=64

Naïve Bayes	Default values

5.2. Quantitative Results for Supervised Machine Learning Algorithms

Different evaluation measures are used to evaluate results produced by supervised and un-supervised algorithms. Accuracy is one of the most important evaluation measure used almost by everyone to check reliability of their results. In data science accuracy means how well the data points are predicted correctly [18]. Below is given the general formula that is followed for accuracy calculation

$$\text{Accuracy} = \frac{\text{No. of correct predictions made}}{\text{Total Prediction made}}$$

In the table 5.2 are shown the accuracy results of supervised machine learning algorithms

Table 5. 2: Supervised algorithms accuracy results – Cross Validation - 1

Supervised Algorithms		
Training data: Self- collected	**Testing data: Publicly available**	
Algorithms	**Training Accuracy**	**Testing Accuracy**
Random Forest	99.12	91.69
SVM	92.59	78.67
KNN	92.14	87.95
GradientBoost	98.6	85.99
AdaBoost	98.87	84.19
Naïve Bayes	82.14	81.65

Best accuracy results are given by random forest giving 99% training accuracy and 91% testing accuracy.

Table 5. 3 : Supervised algorithms accuracy results – Cross Validation - 2

Supervised Algorithms		
Training data: Publicly available	**Testing data: Self Collected**	
Algorithms	**Training Accuracy**	**Testing Accuracy**
Random Forest	99.18	83.8
SVM	96.2	89.5
KNN	96.95	80.94
GradientBoost	99.98	77.05
AdaBoost	98.82	80.94
Naïve Bayes	81.64	84.13

In this case also, premier results are given by random forest.

5.3. Graphical Results for Supervised Machine Learning Algorithms

One of the important classification evaluation techniques is the confusion matrix. A confusion matrix is used over test data to check the number of false and correct predictions done by the model. A Confusion matrix calculates the following

True Positive (TP)

A true positive shows the correct number of predictions of the positive class. For example in this case, if a benign is classified as benign it will be considered as true positive.

True Negative (TN)

A true negative shows the correct no. of predictions of the negative class. For example in this case, if a malware sample is classified as malware it will be considered as true negative.

False Positive (FP)

False-positive shows a negative class sample classified as positive. For example in this case, if a malware sample is classified as benign it will be considered as false positive.

False Negative (FN)

False-negative shows a positive classified as negative. For example in this case, if a benign sample is classified as malware it will be considered as true positive.

Below are shown the confusion matrixes for each of the applied machine learning algorithms using self-collected dataset as training dataset and publicly available dataset as test data:

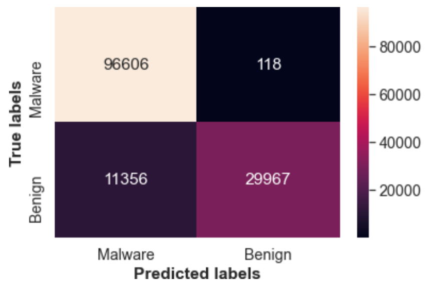


Fig 5. 1: Confusion matrix: Random forest – Cross Validation-1

Large number of samples termed as true positive and slightly better true negative samples can be seen for the random forest; a very little number of samples are classified as true negative.

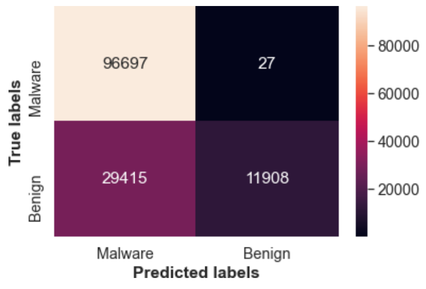


Fig 5. 2: Confusion matrix: SVM – Cross Validation -1

In SVM true positives are even better than random forest performance, but true negatives samples are less in count than random forest

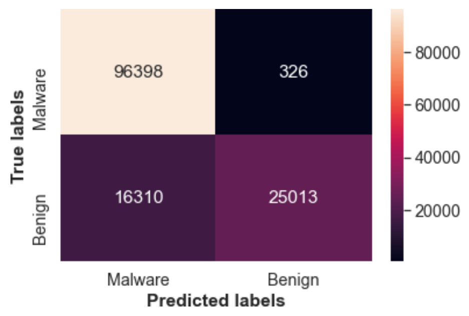


Fig 5. 3: Confusion matrix: KNN – Cross Validation - 1

Both true negative and true positive results are moderate using k-Nearest Neighbours.

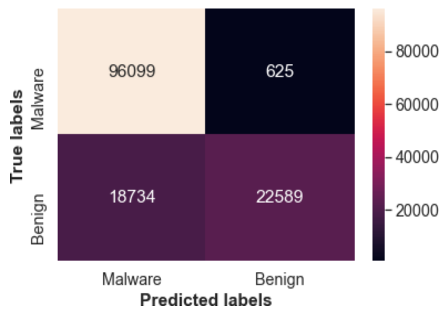


Fig 5. 4: Confusion matrix: Gradient Boosting – Cross Validation -1

Gradient boosting gives better results than KNN.Better true positive and true negative samples are seen than gradient boosting.

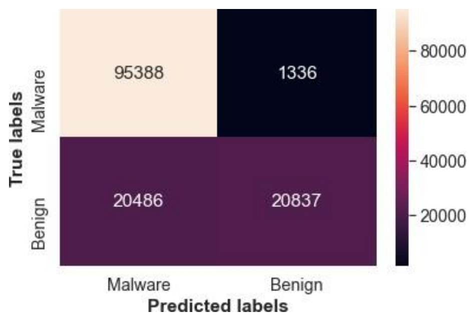


Fig 5. 5: Confusion matrix Adaboost – Cross Validation -1

Much poor results than random forest are given by AdaBoost, given a lower true negative rate than all other models seen before

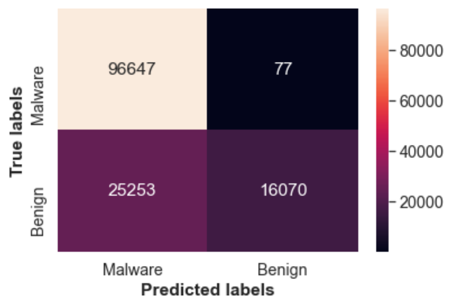


Fig 5. 6: Confusion matrix:Naïve Bayes – Cross Validation-1

Exceptionally good results are given by naïve Bayes in terms of true negatives. Below are shown the confusion matrices while using publicly available dataset for training and self-collected dataset as testing dataset:

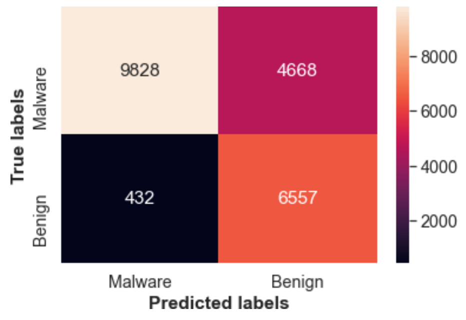


Fig 5. 7: Confusion matrix: Random forest – Cross Validation – 2

Similar results are given by random forest when self-collected data was used as training data.

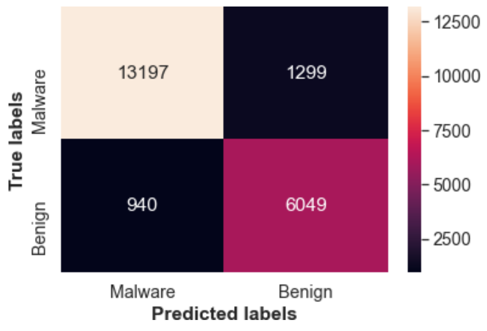


Fig 5. 8: Confusion matrix: SVM – Cross Validation -2

High true negatives seen as well as finest true positive values are given by this model. Much better performance is noticed than using self-collected data as training dataset.

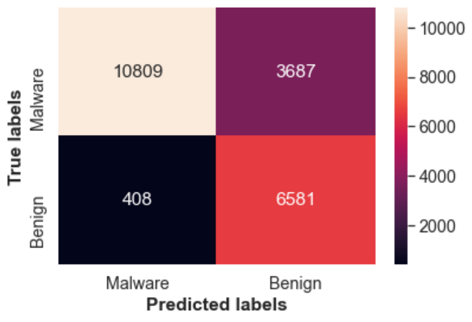


Fig 5. 9: Confusion matrix: KNN – Cross Validation -2

Lower true positives and true negatives as compared to SVM, for KNN using publicly available dataset for testing gave better results than self-collected data as testing dataset.

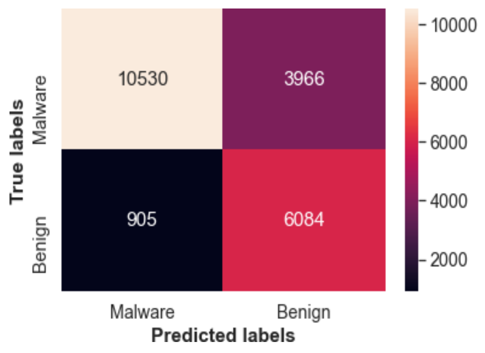


Fig 5. 10: Confusion matrix: Gradiant Boost – Cross Validation -2

High number of false positives and false negative as compared to previous models is given by model of gradient boosting.

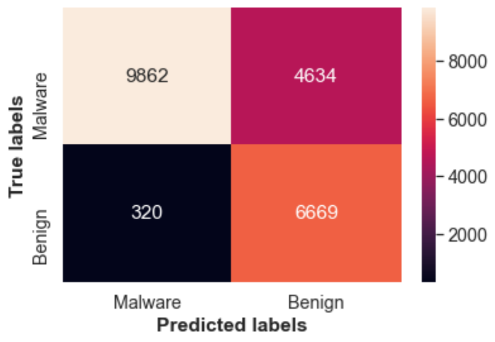


Fig 5. 11: Confusion matrix: AdaBoost – Cross Validation -2

Among all the models previously in via using both types of datasets for training and testing, lowest number of false positives is given by Adaboost.

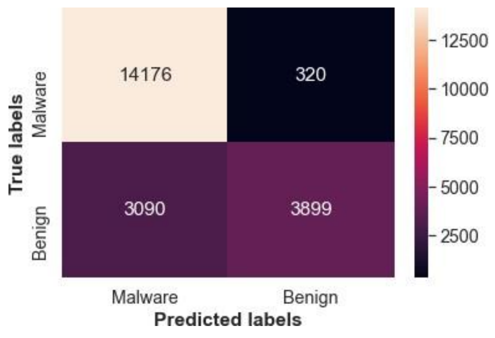


Fig 5. 12: Confusion matrix: Naïve Bayes – Cross Validation - 2

High true positive and true negatives are given by Naïve Bayes

5.4. Classification Reports for Supervised Machine Learning

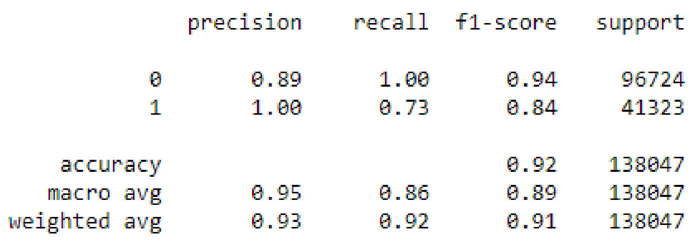
A classification report shows precision, recall and f1-score. Precision provides information about how much of the predications were correct, its value ranges between 0-1, worst results give 0 precision whereas best results give 1.0 precision value. Similarly recall tells how much of the positive class predictions were matched, this also ranges from 0-1 and F1-score tells how many correct positive predictions were done(value range from 0-1)[19] .

$$\text{Precison} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{F1-score} = \frac{2*(\text{Recall}*\ \text{Precision})}{\text{Recall} + \text{Precision}}$$

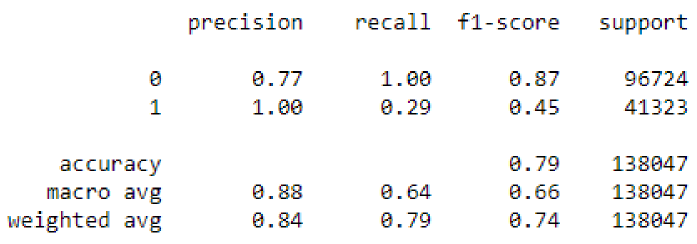
In the figures below are shown the classification reports for the experiments using self-collected dataset as training set and publicly available dataset as test set.



```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     96724
           1       1.00      0.73      0.84     41323

    accuracy                           0.92    138047
   macro avg       0.95      0.86      0.89    138047
weighted avg       0.93      0.92      0.91    138047
```

Fig 5. 13: Classification Report:Random Forest – Cross Validation -1

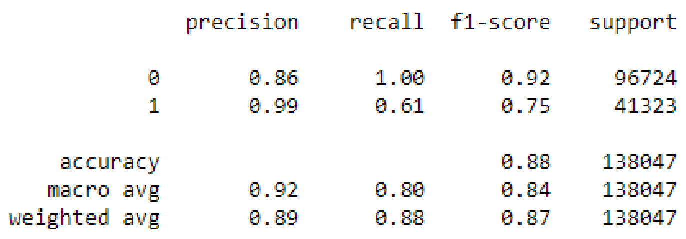
Best results by random forest as very high f1-score,recall, and precision, can be seen for both benign and malware class.



```
              precision    recall  f1-score   support

           0       0.77      1.00      0.87     96724
           1       1.00      0.29      0.45     41323

    accuracy                           0.79    138047
   macro avg       0.88      0.64      0.66    138047
weighted avg       0.84      0.79      0.74    138047
```

Fig 5. 14: Classification Report:SVM – Cross Validation -1

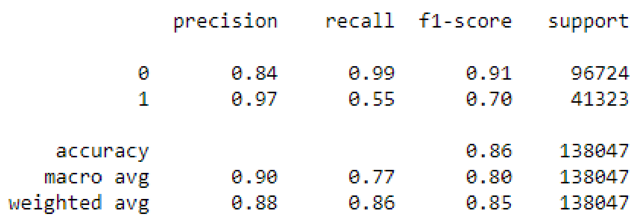
In the case of SVM, poor recall and f1-score is found for the benign class, whereas good results can be seen for malware class



```
              precision    recall  f1-score   support

           0       0.86      1.00      0.92     96724
           1       0.99      0.61      0.75     41323

    accuracy                           0.88    138047
   macro avg       0.92      0.80      0.84    138047
weighted avg       0.89      0.88      0.87    138047
```

Fig 5. 15: Classification Report:KNN – Cross Validation - 1

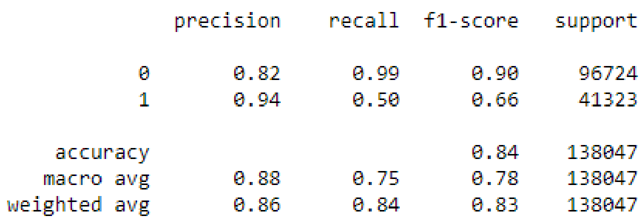
Reasonable results can be seen for malware class, where moderate recall and f1-score are found for the benign class using KNN



```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91     96724
           1       0.97      0.55      0.70     41323

    accuracy                           0.86    138047
   macro avg       0.90      0.77      0.80    138047
weighted avg       0.88      0.86      0.85    138047
```

Fig 5. 16: Classification Report: Gradient Boosting – Cross Validation - 1

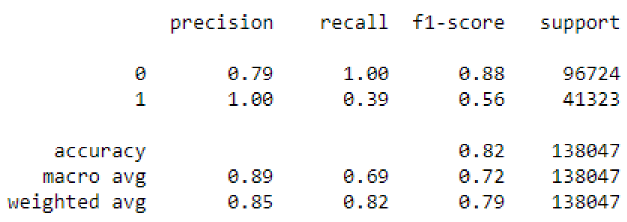
Very good precision is seen for both malware and benign classes recall and F1-score is reasonable for, benign class, using gradient boosting.



```
              precision    recall  f1-score   support

           0       0.82      0.99      0.90     96724
           1       0.94      0.50      0.66     41323

    accuracy                           0.84    138047
   macro avg       0.88      0.75      0.78    138047
weighted avg       0.86      0.84      0.83    138047
```

Fig 5. 17: Classification Report: AdaBoost – Cross Validation -1

Pretty poor results are given by AdaBoost for benign, looking at recall and precision, near to average results are seen whereas good results are achieved for malware class.

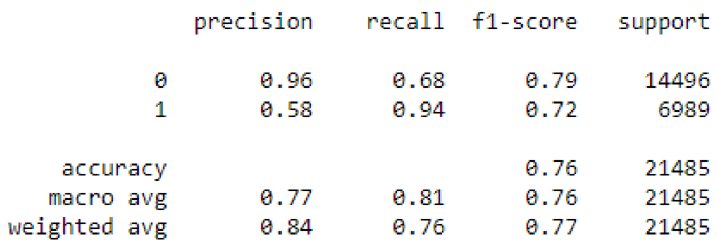


```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88     96724
           1       1.00      0.39      0.56     41323

    accuracy                           0.82    138047
   macro avg       0.89      0.69      0.72    138047
weighted avg       0.85      0.82      0.79    138047
```

Fig 5. 18: Classification Report: Naïve Bayes – Cross Validation -1

Below average results can be seen for recall and f1-score for benign class.

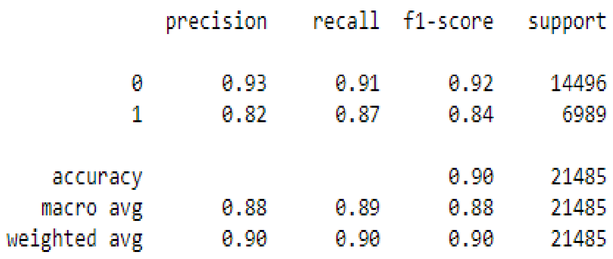
In the figures below are shown the classification reports for the experiments using publicly available dataset as training set and self-collected dataset as test set.



```
              precision    recall  f1-score   support

           0       0.96      0.68      0.79     14496
           1       0.58      0.94      0.72      6989

    accuracy                           0.76     21485
   macro avg       0.77      0.81      0.76     21485
weighted avg       0.84      0.76      0.77     21485
```

Fig 5. 19: Classification Report:Random Forest – Cross Validation -2

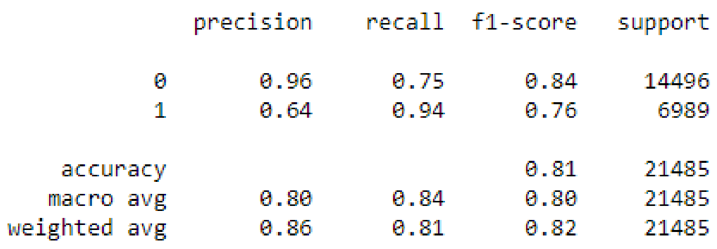
Very high precision and above average f1-score also for malware class.



```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92     14496
           1       0.82      0.87      0.84      6989

    accuracy                           0.90     21485
   macro avg       0.88      0.89      0.88     21485
weighted avg       0.90      0.90      0.90     21485
```

Fig 5. 20: Classification Report:SVM – Cross Validation -2

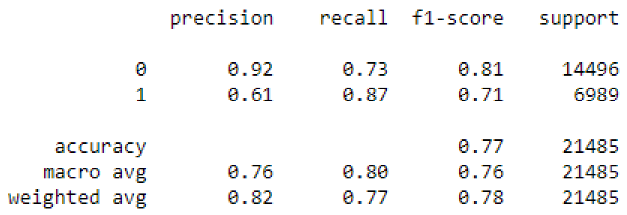
Much better scores are given by SVM while using self-collected data as test dataset as compared to using publicly available dataset for testing.



```
              precision    recall  f1-score   support

           0       0.96      0.75      0.84     14496
           1       0.64      0.94      0.76      6989

    accuracy                           0.81     21485
   macro avg       0.80      0.84      0.80     21485
weighted avg       0.86      0.81      0.82     21485
```

Fig 5. 21: Classification Report:KNN – Cross Validation -2

For malware class F1-score as well as other scores are very reliable as compared to previous models utilized.



```
              precision    recall  f1-score   support

           0       0.92      0.73      0.81     14496
           1       0.61      0.87      0.71      6989

    accuracy                           0.77     21485
   macro avg       0.76      0.80      0.76     21485
weighted avg       0.82      0.77      0.78     21485
```

Fig 5. 22: Classification Report:Gradient Boosting – Cross Validation -2

High precision score for malware class and average F1- scores for both the classes.

```
              precision    recall  f1-score   support

           0       0.97      0.68      0.80     14496
           1       0.59      0.95      0.73      6989

    accuracy                           0.77     21485
   macro avg       0.78      0.82      0.76     21485
weighted avg       0.85      0.77      0.78     21485
```

Fig 5. 23: Classification Report:AdaBoost – Cross Validation -2

For both classes fine recall and f1-score whereas very high recall for benign class.

```
              precision    recall  f1-score   support

           0       0.82      0.98      0.89     14496
           1       0.92      0.56      0.70      6989

    accuracy                           0.84     21485
   macro avg       0.87      0.77      0.79     21485
weighted avg       0.85      0.84      0.83     21485
```

Fig 5. 24: Classification Report:Naïve Bayes – Cross Validation -2

5.5.ROC curve for Supervised Machine Learning Algorithms

In ROC curve, this explains how much good the model distinguishes between both classes. If the curve is closer to top left corner, the model has performed very well. In the figure shown below the ROC curve for supervised algorithms

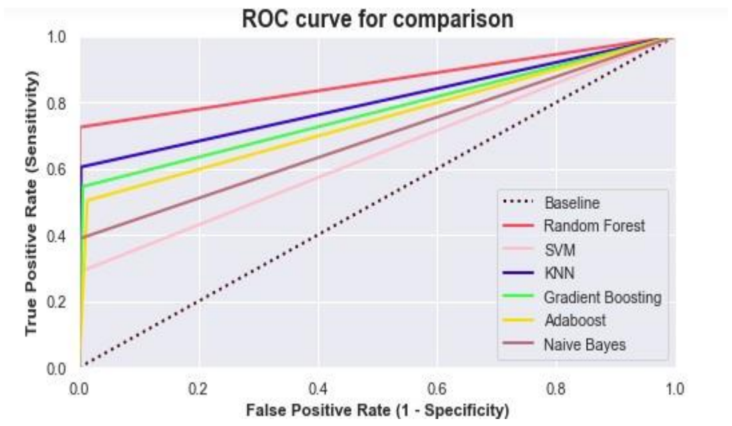


Fig 5. 25: ROC curve for supervised algorithms- Cross Validation -1

In the figure shown above it can be seen that random forest gives the best result, whereas SVM gives the worst results.

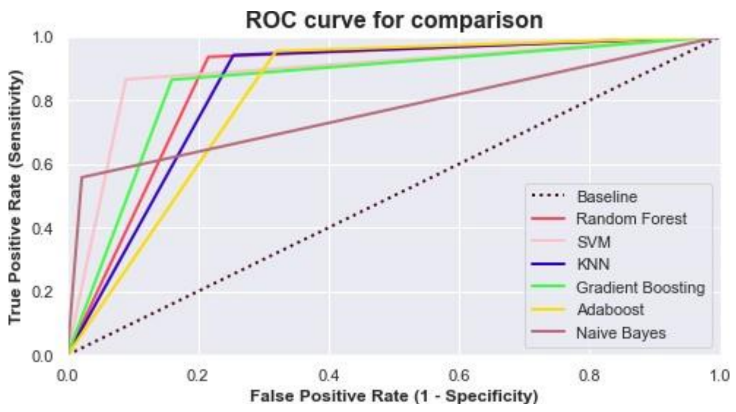


Fig 5. 26: ROC curve for supervised algorithms- Cross Validation -2

Similar results as in previous cross validation phase are seen in this ROC curve as well.

5.6. Quantitative Results for Un-Supervised Machine Learning Algorithms

ACC in the results is the short term used for accuracy.F1-score; precision and recall are already explained. AUC means the area under the curve, it shows how much the model is capable of distinguishing between the two classes. AUC having a score of 0.5 means no discrimination ability seen by the model. A score of 0.6-0.7 is termed as poor discrimination more the value is closer to 1.0; the better is the discrimination ability of the model [20].

In the tables below are the results of training and testing using all the features

Table 5. 4: K-Means results, using all 36 features

K-Means Training Results – using all 36 features				
ACC	**AUC**	**F1 - Score**	**Precision**	**Recall**
86.25	0.861	0.788	0.862	0.862
K-Means Testing Results – using all 36 features				
ACC	**AUC**	**F1 - Score**	**Precision**	**Recall**
72.13	0.763	0.673	0.721	0.797

Average values for AUC, f1-score, precision and recall can be seen on test data with k-means using all features whereas very good training results are seen.

Table 5. 5: K-Means-Mini results, using all 36 features

K-Means-Mini Training Results – using all 36 features				
ACC	**AUC**	**F1 - Score**	**Precision**	**Recall**
86.25	0.861	0.788	0.873	0.862
K-Means-Mini Testing Results – using all 36 features				
ACC	**AUC**	**F1 - Score**	**Precision**	**Recall**
72.14	0.763	0.673	0.798	0.721

Results similar to k-means are seen with k-means–mini with better precision value upon testing dataset as well as the training dataset. Whereas recall value dropped over the testing dataset

Table 5. 6: GMM results, using all 36 features

GMM Training Results – using all 36 features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.47	0.831	0.755	0.851	0.844
GMM Testing Results – using all 36 features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
72.71	0.728	0.635	0.755	0.727

Similar results as given by k-means and k-means-mini as given by GMM using all 36 features.

Table 5. 7: Fuzzy c-menas, using all 36 features

Fuzzy c-means Training Results – using all 36 features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.99	0.147	0.090	0.226	0.150
Fuzzy c-means Testing Results – using all 36 features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.01	0.765	0.692	0.840	0.854

Very good precision and recall values are given by fuzzy c-means using all 36 features, much better than the other models used before.

Table 5. 8: K-Means results, with PCA features

K-Means Training Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.021	0.86	0.789	0.873	0.864
K-Means Testing Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.02	0.765	0.692	0.854	0.840

Instead of using all the features, using the approach of dimensionality reduction and using only 3 features reduced using PCA, much better results are given by k-means, precision and recall has increased from 0.7 to 0.8 only by using dimensionality reduction.

Table 5. 9: K-Means-Mini results, with PCA features

K-Means-Mini Training Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
81.49	0.691	0.553	0.814	1.0
K-Means-Mini Testing Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
83.21	0.753	0.671	0.832	0.847

Just like k-means using PCA results for k-means-mini clustering better results are seen than using k-means-mini with all the features. Precision and Recall values have been improved using PCA.

Table 5. 10: GMM results, with PCA features

GMM Training Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.54	0.841	0.762	0.845	0.856
GMM Testing Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.02	0.765	0.629	0.840	0.854

Better testing and training results are given by the gaussian mixture model using PCA features in terms of precision and recall.

Table 5. 11: Fuzzy c-means results, with PCA features

Fuzzy c-means Training Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
85.09	0.147	0.090	0.149	0.225
Fuzzy c-means Testing Results – with PCA features				
ACC	**AUC**	**F1 – Score**	**Precision**	**Recall**
84.02	0.765	0.692	0.840	0.854

Best results were given by the gaussian mixture model either with all the features or using PCA specific features. Results were improved after using dimensionality reduction.

5.7. Result Analysis of Machine Learning Algorithms

Moving towards the result analysis of supervised algorithms, it is observed that random-forest gives most reliable results, using roc random forest is the closest to top left corner proving its distinguishing ability between malware and benign samples. On the other hand, SVM shows most unreliable prediction upon the dataset of malware and benign samples whereas gradient boosting doesn't performs well on self-collected dataset upon testing.

In un-supervised algorithms, better results were deduced after doing dimensionality reduction using PCA. Reliable clustering results were produced by all the unsupervised algorithms. Only fuzzy c-means didn't perform as expected. In both supervised and unsupervised algorithms, models were well generalized and no overfitting could be seen any of the models

5.8. AutoEncoders Parameters

In the table next are summarized the parameters of autoencoder for first models which clusters using k-means only.

Table 5. 12: Parameters for Model-1

Parameters – Model 1	
Batch Size	32
Epochs	100
Learning Rate	Default
Loss Function	Binary cross entropy
Momentum weight	Default
Optimizer	Adam

In the table 5.13 below are shown the parameters used for model-2 with publicly available dataset:

Table 5. 13:Parameters for model-2

Parameters autoencoder			
Model-2		**Model-3**	
Batch Size	32	**Batch Size**	32
Epochs	100	**Epochs**	50
Learning Rate	Default	**Learning Rate**	Default
Loss Function	Binary Cross entropy	**Loss Function**	Binary Cross entropy
Momentum weight	Default	**Momentum weight**	Default
Optimizer	Adam	**Optimizer**	Adam

5.9. Clustering results for Static features with Dimensionality reduction

Below in table 5.14 are shown the results of k-Means clustering using all features, k-means after passing dataset through PCA and k-means clustering results with autoencoder

Table 5. 14: K-Means clustering results

Accuracy	**K-Means**	**K-Means+PCA**	**K-Means+ AE**
	86.4%	86.7%	90.3%

ACC, recall, precision, and f1-score are explained previously. Rand Index measures the resemblance between the two clusters. A value closer to 1.0 means a good rand

index [66]. Jaccard Index is similar to the rand index with few differences in the approach used for measuring similarity between clusters.
Below in table 5.15 are shown the results of k-Means, GMM, fuzzy c-means results after clustering from data passed through autoencoder.

Table 5. 15: K-Means, GMM, fuzzy c-means clustering results with encoder only

Algorithm	**ACC**	**Precision**	**Recall**	**F1-score**	**Rand Index**	**Jaccard**
K-Means	86.0	0.875	0.857	0.892	0.505	0.806
GMM	88.6	0.887	0.886	0.919	0.586	0.849
FCM	85.1	0.871	0.851	0.888	0.489	0.798

Best results are given by Gaussian mixture model, in terms of all the evaluation metrics used, f1-score of 0.9 shows that the model has performed very well in terms of clustering even precision and recall values have escalated using this approach.

Below in table 5.16 are shown the results of k-Means, GMM, fuzzy c-means results after clustering from data passed through ensemble based autoencoder.

Table 5. 16: Ensemble based autoncoder : K-Means, GMM, fuzzy c-means clustering results

Algorithm	**ACC**	**Precision**	**Recall**	**F1-score**	**Rand Index**	**Jaccard**
K-Means	95.0	0.946	0.947	0.962	0.792	0.927
GMM	95.39	0.954	0.954	0.967	0.82	0.936
FCM	94.8	0.948	0.948	0.963	0.798	0.929

Even better results are given by the Gaussian mixture model by using only the encoder portion of the autoencoder, as the latent space of the autoencoder is used it contains the most important features information, after that this information is parsed through ensemble block which refine the features even more, hence best results are produced using this approach.

5.9. Confusion matrix for clustering with Encoder only and ensemble based Autoencoder

In the below given confusion matrices a comparison is done between results of clustering using encoder portion only and clustering with ensemble based autoencoder.

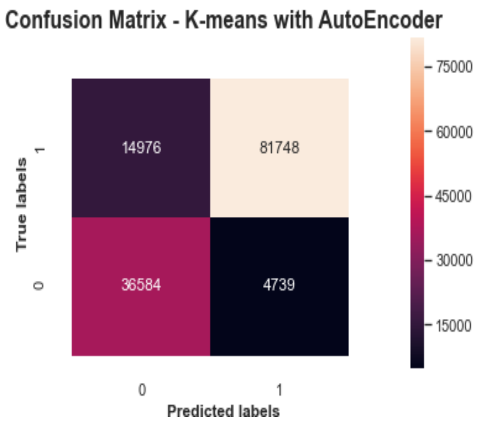


Fig 5. 27: Confusion matrix: K-Means clustering with encoder only

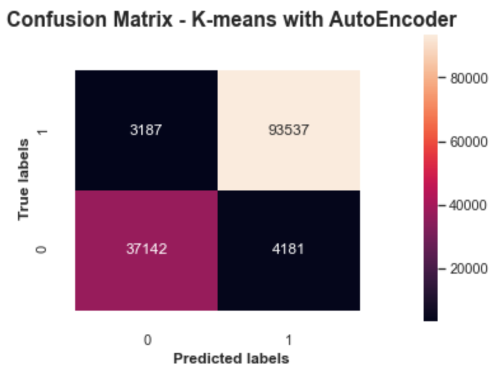


Fig 5. 28: Confusion matrix: K-Means with ensemble based autoencoder

Looking at the above two confusion matrix the true positives and true negatives given by ensemble NN based autoencoder are very much more reliable from the true positives and true negatives given by using encoder portion only.

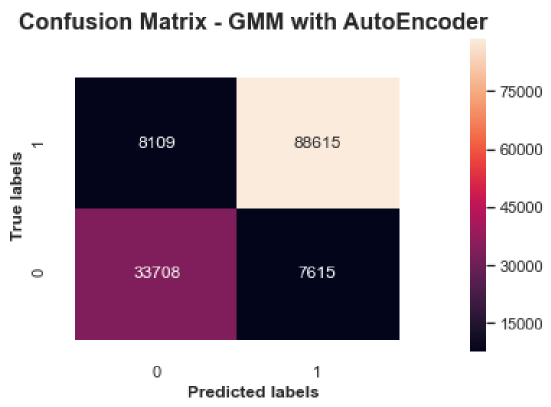


Fig 5. 29: Confusion matrix: GMM clustering with encoder only

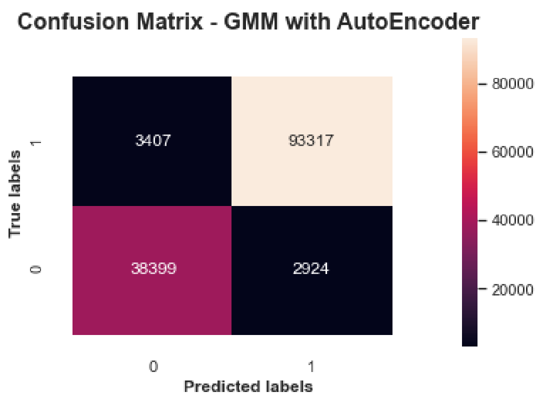


Fig 5. 30 : Confusion matrix: GMM with ensemble NN based autoencoder

As, expected results given by Gaussian mixture model are better than those given by k-means, also the results of ensemble NN based autoencoder are much better than those given by only encoder.

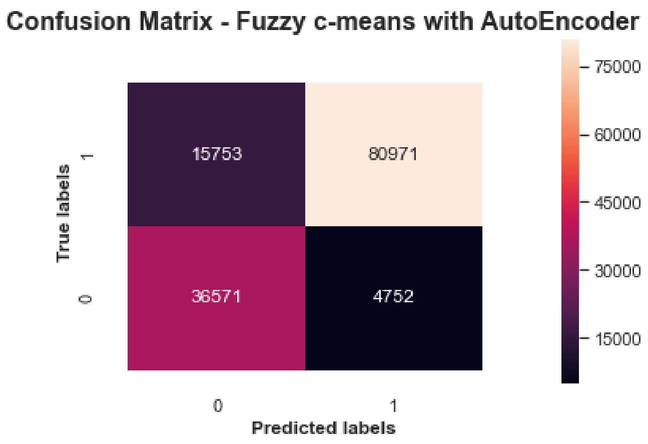


Fig 5. 31: Confusion matrix: Fuzzy c-means clustering with encoder only

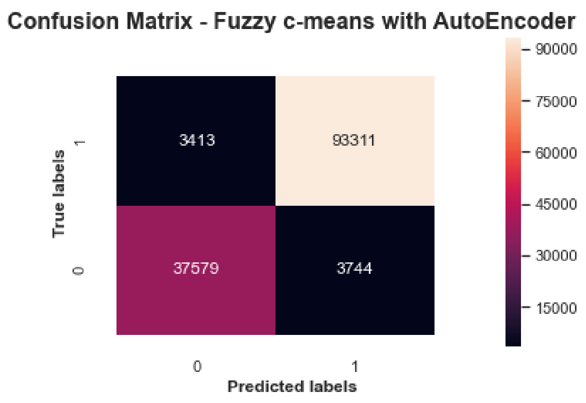


Fig 5. 32: Confusion matrix: Fuzzy c-means with ensemble NN based autoencoder

So far, fuzzy c-means gives lowest results as compared to k-means and Gaussian mixture model, but also in this case ensemble based autoencoder gave much better true positives and true negatives.

5.10. ROC curve : Autoencoders

Below is shown the ROC curve for clustering with encoder only and ROC curve for ensemble NN based autoencoder.

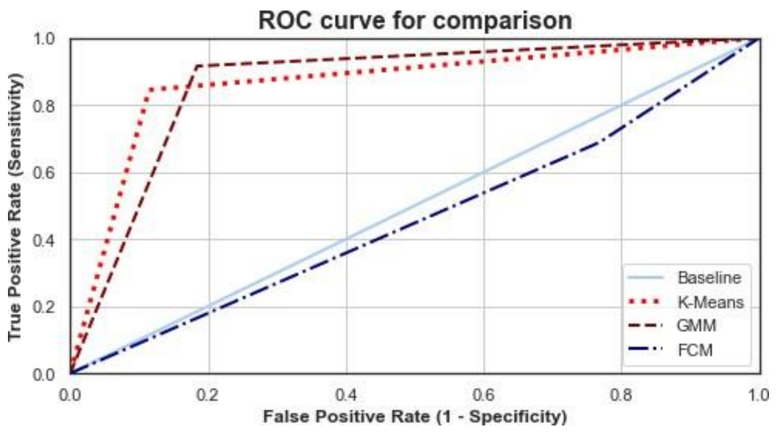


Fig 5. 33: ROC curve for clustering based upon encoder

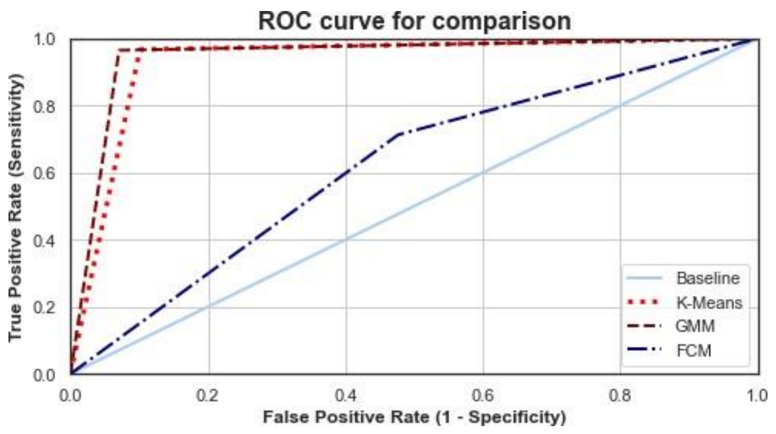


Fig 5. 34 : ROC curve for clustering based upon ensemble NN based autoencoder

In both the cases, best performance is given by Gaussian mixture model.

Summary

The results of methodologies demonstrated in the previous chapter are discussed. Quantitative results are shared as well as evaluation metrics of the confusion matrix and for comparison of multiple algorithms ROC curve is used.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1. Conclusion

In this study usage of static features for classification is demonstrated briefly. Using this technique machine learning algorithms yielded good results but very convincing results were given by deep learning especially when the classification was done with an ensemble-based autoencoder for dimensionality reduction. All six algorithms gave training and testing accuracy above 80% but outstanding results were given by dimensionality reduction approaches. After performing clustering with PCA and autoencoder the f1-score in most of the cases sored up to nearly 0.8-0.9, which is termed as outstanding results. Even with machine learning, using the approach of cross-validation across a self-collected dataset and publicly available dataset a robust model was created. Moreover, the authenticity of the self-collected dataset can be seen after training machine learning algorithms on this dataset and testing the models with the publicly available datasets. On the other hand for unsupervised results using a self-collected dataset for testing again proved its authenticity.

6.2. Recommendations

1) Collecting more dataset will give the models more strength for training.
2) Time efficient data collection techniques to be used.
3) Dynamic analysis, memory analysis, polymorphic behavior of malwares should be incorporated in order to make it a real time application to detect malwares.
4) GAN's should be used over static, dynamic and memory analysis datasets.

6.3. Future Work

First thing in future work is to collect more data and not only static features but also heterogeneous features including dynamic and memory analysis based data. More un-supervised algorithms to be incorporated such as agglomerative clustering etc. One of the tasks to be included in future work would be to train generative adversarial networks upon the collected dataset of static, dynamic, and memory analysis. Goals would be to create a zero-day attack detector based upon GAN’s. Other than that in case of unsupervised dimensionality reduction based clustering, fuzzy c-means to be extended to fuzzy hashing, etc.

REFERENCES

[1] "Malware Law and Legal Definition | USLegal, Inc." Feb. 26, 2020, [Online]. Available: https://definitions.uslegal.com/m/malware/.

[2] "ISS World Hit with Malware Attack that Shuts Down Global Computer Network." Feb. 28, 2020, [Online]. Available: https://threatpost.com/iss-world-hit-with-malware-attack-that-shuts-down-global-computer-network/153109/.

[3] "Report: Malware Poisons One-Third of World's Computers | Malware | TechNewsWorld." Feb. 28, 2020, [Online]. Available: https://www.technewsworld.com/story/80707.html.

[4] R. Benzmüller, "In 2017 every 4.2 seconds a new malware specimen emerges." Feb. 28, 2017, [Online]. Available: https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017.

[5] "Pakistan at high risk of malware attack, says Microsoft Security Report," *TechJuice*. Feb. 2017, [Online]. Available: https://www.techjuice.pk/pakistan-high-risk-malware-attack-microsoft-security-report/.

[6] P. Staff, "One in Four Computers in Pakistan are Attacked by Malware: Microsoft." Feb. 29, 2020, [Online]. Available: https://propakistani.pk/2017/09/19/one-four-computers-pakistan-attacked-malware-microsoft/.

[7] W. Zamora, "What's the difference between antivirus and anti-malware?," *Malwarebytes Labs*. Feb. 2015, [Online]. Available: https://blog.malwarebytes.com/101/2015/09/whats-the-difference-between-antivirus-and-anti-malware/.

[8] "McAfee Downloads - Antivirus, Antimalware, Virus Scan, Free Trials | McAfee Products." Feb. 29, 2020, [Online]. Available: https://www.mcafee.com/enterprise/en-us/downloads.html.

[9] "Total Protection for Your Devices & Files - TotalAV.com." Feb. 29, 2020, [Online]. Available: https://www.totalav.com/.

[10] "Antivirus and Malware Protection | Antivirus for PC, Mac®, Android^{TM} & iOS." Feb. 29, 2020, [Online]. Available: https://us.norton.com/antivirus.

[11] "Malwarebytes Cybersecurity for Home and Business," *Malwarebytes*. Feb. 29, 2020, [Online]. Available: https://www.malwarebytes.com/.

[12] A. Mujumdar, G. Masiwal, and D. B. B. Meshram, "Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches," vol. 2, no. 6, p. 3, [Online]. Available: http://ijarcet.org/wp-content/uploads/VOLUME-2-ISSUE-6-2037-2039.pdf.

[13] “What is Zero Day Exploit? | Definition and Risks | Kaspersky.” Mar. 02, 2020, [Online]. Available: https://www.kaspersky.com/resource-center/definitions/zero-day-exploit.

[14] “What’s a ‘Zero-Day’ Attack?,” *Ask Leo!* Mar. 02, 2017, [Online]. Available: https://askleo.com/whats-zero-day-attack/.

[15] “What is Malware Analysis? Defining and Outlining the Process of Malware Analysis | Digital Guardian.” https://digitalguardian.com/blog/what-malware-analysis-defining-and-outlining-process-malware-analysis (accessed Jun. 13, 2020).

[16] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, “Malware detection using machine learning and deep learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11297 LNCS, pp. 402–411, 2018, doi: 10.1007/978-3-030-04780-1_28.

[17] K. O. Babaagba and S. O. Adesanya, “A study on the effect of feature selection on malware analysis using machine learning,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1481, pp. 51–55, 2019, doi: 10.1145/3318396.3318448.

[18] R. Sihwail, K. Omar, K. A. Z. Ariffin, and S. Al Afghani, “Malware detection approach based on artifacts in memory image and dynamic analysis,” *Appl. Sci.*, vol. 9, no. 18, 2019, doi: 10.3390/app9183680.

[19] K. Sethi, R. Kumar, L. Sethi, P. Bera, and P. K. Patra, “A novel machine learning based malware detection and classification framework,” *2019 Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2019*, pp. 1–4, 2019, doi: 10.1109/CyberSecPODS.2019.8885196.

[20] M. Ijaz, M. H. Durad, and M. Ismail, “Static and Dynamic Malware Analysis Using Machine Learning,” *Proc. 2019 16th Int. Bhurban Conf. Appl. Sci. Technol. IBCAST 2019*, pp. 687–691, 2019, doi: 10.1109/IBCAST.2019.8667136.

[21] C. M. Chen, S. H. Wang, D. W. Wen, G. H. Lai, and M. K. Sun, “Applying Convolutional Neural Network for Malware Detection,” *2019 IEEE 10th Int. Conf. Aware. Sci. Technol. iCAST 2019 - Proc.*, pp. 1–5, 2019, doi: 10.1109/ICAwST.2019.8923568.

[22] A. I. Elkhawas and N. Abdelbaki, “Malware Detection using Opcode Trigram Sequence with SVM,” *2018 26th Int. Conf. Software, Telecommun. Comput. Networks, SoftCOM 2018*, pp. 7–11, 2018, doi: 10.23919/SOFTCOM.2018.8555738.

[23] A. Irshad, R. Maurya, M. K. Dutta, R. Burget, and V. Uher, “Feature optimization for run time analysis of malware in windows operating system using machine learning

approach," *2019 42nd Int. Conf. Telecommun. Signal Process. TSP 2019*, pp. 255–260, 2019, doi: 10.1109/TSP.2019.8768808.

[24] B. M. Khammas, S. Hasan, R. A. Ahmed, J. S. Bassi, and I. Ismail, "Accuracy Improved Malware Detection Method using Snort Sub-signatures and Machine Learning Techniques," *2018 10th Comput. Sci. Electron. Eng. Conf. CEEC 2018 - Proc.*, pp. 107–112, 2019, doi: 10.1109/CEEC.2018.8674233.

[25] Z. Fuyong and Z. Tiezhu, "Malware detection and classification based on n-grams attribute similarity," *Proc. - 2017 IEEE Int. Conf. Comput. Sci. Eng. IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput. CSE EUC 2017*, vol. 1, pp. 793–796, 2017, doi: 10.1109/CSE-EUC.2017.157.

[26] S. C. Hsiao, D. Y. Kao, and R. Tso, "Malware-detection model using learning-based discovery of static features," *2018 IEEE Conf. Appl. Inf. Netw. Secur. AINS 2018*, pp. 54–59, 2019, doi: 10.1109/IISA.2018.8631505.

[27] S. A. Roseline and S. Geetha, "Intelligent Malware Detection using Oblique Random Forest Paradigm," *2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018*, pp. 330–336, 2018, doi: 10.1109/ICACCI.2018.8554903.

[28] G. Pitolli, L. Aniello, G. Laurenza, L. Querzoni, and R. Baldoni, "Malware family identification with BIRCH clustering," *Proc. - Int. Carnahan Conf. Secur. Technol.*, vol. 2017-Octob, pp. 1–6, 2017, doi: 10.1109/CCST.2017.8167802.

[29] S. Abijah Roseline, A. D. Sasisri, S. Geetha, and C. Balasubramanian, "Towards efficient malware detection and classification using multilayered random forest ensemble technique," *Proc. - Int. Carnahan Conf. Secur. Technol.*, vol. 2019-Octob, 2019, doi: 10.1109/CCST.2019.8888406.

[30] R. C. Damale, "System Using Machine Learning Algorithms," no. Iciccs, pp. 414–419, 2018.

[31] N. Udayakumar, V. J. Saglani, A. V. Cupta, and T. Subbulakshmi, "Malware Classification Using Machine Learning Algorithms," *Proc. 2nd Int. Conf. Trends Electron. Informatics, ICOEI 2018*, pp. 1007–1012, 2018, doi: 10.1109/ICOEI.2018.8553780.

[32] C. H. Kim, E. K. Kabanga, and S. J. Kang, "Classifying malware using convolutional gated neural network," *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2018-Febru, pp. 40–44, 2018, doi: 10.23919/ICACT.2018.8323640.

[33] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," *Proc. ACMSE 2018 Conf.*, vol. 2018-Janua, 2018, doi: 10.1145/3190645.3190692.

[34] R. U. Khan, X. Zhang, and R. Kumar, “Analysis of ResNet and GoogleNet models for malware detection,” *J. Comput. Virol. Hacking Tech.*, vol. 15, no. 1, pp. 29–37, 2019, doi: 10.1007/s11416-018-0324-z.

[35] K. He and D. S. Kim, “Malware detection with malware images using deep learning techniques,” *Proc. - 2019 18th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. IEEE Int. Conf. Big Data Sci. Eng. Trust. 2019*, pp. 95–102, 2019, doi: 10.1109/TrustCom/BigDataSE.2019.00022.

[36] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, “Malware Classification with Deep Convolutional Neural Networks,” *2018 9th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2018 - Proc.*, vol. 2018-Janua, pp. 1–5, 2018, doi: 10.1109/NTMS.2018.8328749.

[37] D. Kornish, J. Geary, V. Sansing, S. Ezekiel, L. Pearlstein, and L. Njilla, “Malware Classification using Deep Convolutional Neural Networks,” *Proc. - Appl. Imag. Pattern Recognit. Work.*, vol. 2018-Octob, pp. 1–6, 2018, doi: 10.1109/AIPR.2018.8707429.

[38] W. W. Lo, X. Yang, and Y. Wang, “An xception convolutional neural network for malware classification with transfer learning,” *2019 10th IFIP Int. Conf. New Technol. Mobil. Secur. NTMS 2019 - Proc. Work.*, pp. 1–5, 2019, doi: 10.1109/NTMS.2019.8763852.

[39] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. De Geus, “Malicious software classification using transfer learning of ResNet-50 deep neural network,” *Proc. - 16th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2017*, vol. 2017-Decem, pp. 1011–1014, 2017, doi: 10.1109/ICMLA.2017.00-19.

[40] S. Akarsh, K. Simran, P. Poornachandran, V. K. Menon, and P. K. Soman, “Deep Learning Framework and Visualization for Malware Classification,” *2019 5th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2019*, pp. 1059–1063, 2019, doi: 10.1109/ICACCS.2019.8728471.

[41] “Hauri Co., Ltd.” https://www.hauri.co.kr/ (accessed Jun. 15, 2020).

[42] “KAIST Cyber Security Research Center.” https://csrc.kaist.ac.kr/ (accessed Jun. 15, 2020).

[43] В. А. К. Российской and А. Наук, “mlimdl,” vol. 12, no. 1, p. 145, 2017.

[44] W. Jun-Ling and W. Shuo-Hao, “Malicious Classification Based on Deep Learning and Visualization,” *2019 2nd Int. Conf. Artif. Intell. Big Data, ICAIBD 2019*, pp. 223–228, 2019, doi: 10.1109/ICAIBD.2019.8837017.

[45] “Amazon.com: Mastering Machine Learning for Penetration Testing: Develop an extensive skill set to break self-learning systems using Python eBook: Chebbi, Chiheb: Kindle Store.” https://www.amazon.com/Mastering-Machine-Learning-Penetration-Testing-ebook/dp/B07CSN7QQ1 (accessed Jun. 25, 2020).

[46] “Amazon.com: Advanced Infrastructure Penetration Testing: Defend your systems from methodized and proficient attackers eBook: Chebbi, Chiheb: Kindle Store.” https://www.amazon.com/Advanced-Infrastructure-Penetration-Testing-methodized-ebook/dp/B076QC8FRT (accessed Jun. 25, 2020).

[47] “Chiheb Chebbi.” https://www.packtpub.com/authors/chiheb-chebbi (accessed Jun. 25, 2020).

[48] “GitHub - PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing: Mastering Machine Learning for Penetration Testing, published by Packt.” https://github.com/PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing (accessed Jun. 25, 2020).

[49] “VirusShare.com.” https://virusshare.com/ (accessed Jun. 25, 2020).

[50] “Free Software Downloads and Reviews for Windows, Android, Mac, and iOS – CNET Download.” https://download.cnet.com/ (accessed Jun. 25, 2020).

[51] “VirusShare.com.” https://virusshare.com/about.4n6 (accessed Jun. 25, 2020).

[52] “Download VMware Workstation Pro.” https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html (accessed Jun. 25, 2020).

[53] “Download Ubuntu Desktop | Download | Ubuntu.” https://ubuntu.com/download/desktop (accessed Jun. 25, 2020).

[54] “Project Jupyter | Installing the Jupyter Software.” https://jupyter.org/install (accessed Jun. 25, 2020).

[55] “PE File Archives - PE File.” http://pefile.net/category/pe-file/ (accessed Jun. 17, 2020).

[56] “Portable Executable and Common Object File Format Specification 4.1.” http://bytepointer.com/resources/pecoff_v4.1.htm (accessed Jun. 17, 2020).

[57] “Metadata Tables - 1. The PE File Format.” http://www.vijaymukhi.com/documents/books/metadata/chap1.htm (accessed Jun. 17, 2020).

[58] “/VERSION (Version Information) | Microsoft Docs.” https://docs.microsoft.com/en-us/cpp/build/reference/version-version-information?view=vs-2019 (accessed Jun. 17,

2020).

[59] “Recent Advances in Intrusion Detection: 12th International Symposium, RAID ... - Google Books.” https://books.google.com.pk/books?id=txpuCQAAQBAJ&pg=PA126&lpg=PA126&dq=size+of+optional+header+for+benign+file&source=bl&ots=2JHhmqniT6&sig=ACfU3U0zfmu5LbVL-1cPSYJneLG5Sv3FeA&hl=en&sa=X&ved=2ahUKEwiloIKVjZLoAhUHkxQKHdPCCnsQ6AEwA3oECAoQAQ#v=onepage&q=size of optional header for benign file&f=false (accessed Jun. 18, 2020).

[60] “IMAGE_OPTIONAL_HEADER32 (winnt.h) - Win32 apps | Microsoft Docs.” https://docs.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-image_optional_header32?redirectedfrom=MSDN (accessed Jun. 18, 2020).

[61] “Portable Executable File Format.” https://blog.kowalczyk.info/articles/pefileformat.html (accessed Jul. 07, 2020) .

[62] C. K. Ng, F. Jiang, L. Y. Zhang, and W. Zhou, “Static malware clustering using enhanced deep embedding method,” *Concurr. Comput.* , vol. 31, no. 19, pp. 1–16, 2019, doi: 10.1002/cpe.5234.

[63] “Choosing Evaluation Metrics.” https://predictionio.apache.org/evaluation/metricchoose/ (accessed Jul. 08, 2020).

[64] “Understanding a Classification Report For Your Machine Learning Model.” https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397 (accessed Jul. 08, 2020).

[65] “What is the value of the area under the roc curve (AUC) to conclude that a classifier is excellent?” https://www.researchgate.net/post/What_is_the_value_of_the_area_under_the_roc_curve_AUC_to_conclude_that_a_classifier_is_excellent (accessed Jul. 08, 2020).

[66] “Rand index - Wikipedia.” https://en.wikipedia.org/wiki/Rand_index (accessed Jul. 08, 2020).