DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

in

Bachelor of Technology
COMPUTER SCIENCE AND ENGINEERING

Major Project Phase-II Report

(CLOUD BASED MALWARE DETECTION SYSTEM)

By
Vinay J - ENG18CS0322
Rohit K - ENG18CS0232
Zuhair Bilal - ENG18CS0332
Syed Ali Zuhaib - ENG19CS1021

Under the supervision of

Dr Mouleeswaran SK
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY
(2021-2022)

DAYANANDA SAGAR UNIVERSITY

School of Engineering
Department of Computer Science & Engineering
Kudlu Gate, Bangalore – 560068
Karnataka, India

CERTIFICATE

This is to certify that the Phase-II project work titled **"CLOUD BASED MALWARE DETECTION SYSTEM"** is carried out by **ROHIT K (ENG18CS0232), VINAY J (ENG18CS322), ZUHAIR BILAL (ENG18CS0332), SYED ALI ZUHAIB (ENG19CS1021),** a bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2021-2022**.

Dr. Mouleeswaran S K	**Dr Girisha G S**	**Dr. A Srinivas**
Associate Professor, Dept. of CSE School of Engineering Dayananda Sagar University Date:	Chairman CSE School of Engineering Dayananda Sagar University Date:	Associate School of Engineering Dayananda Sagar University Date:

Name of the Examiner **Signature of Examiner**

1.
2.

DECLARATION

We, **ROHIT K (ENG18CS0232), VINAY J (ENG18CS0322), ZUHAIR BILAL (ENG18CS0332), SYED ALI ZUHAIB (ENG19CS1021),** are students of the eigth semester B.Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the phase-II project titled **"CLOUD BASED MALWARE DETECTION SYSTEM"** has been carried out by us and submitted in partial fulfillment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2021-2022**.

Student **Signature**

Name1: Rohit K
USN : ENG18CS0232

Name2: Vinay J
USN : ENG18CS0322

Name3: Zuhair Bilal
USN : ENG18CS0332

Name4: Syed Ali Zuhaib
USN : ENG19CS1021

Place : Bangalore
Date :

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. A Srinivas. Dean**, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman**, **Computer Science, and Engineering**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible.

We would like to thank our guide **Dr. Mouleeswaran S K**, **Associate Professor**, **Dept. of Computer Science and Engineering**, **Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and the fruitful culmination of the project.

We would like to thank our Project Coordinator Dr. Meenakshi Malhotra and Dr. Bharanidharan N, and all the staff members of Computer Science and Engineering for their support.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Project work

ROHIT K - ENG18CS0232
VINAY J - ENG18CS0322
ZUHAIR BILAL - ENG18CS0332
SYED ALI ZUHAIB - ENG19CS1021

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS

PE	Portable Executable
ML	Machine Learning
OS	Operating System
AWS	Amazon Web Services

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The aim is to utilize the concept of machine learning and to build a model using ensemble algorithm, which can be trained efficiently to detect malwares in a system. The aim of this project is to develop a Machine Learning Model, which is able to classify whether a PE file is legitimate or malicious with the help of supervised machine learning algorithms as there is a continuous rise in the computer technology, the hackers and the hacking incidents are increasing. Because of which the demand of security is also elevating. Malwares have been a great pain for the computer users around the world. Huge organizations have gone into huge losses due to this loophole in security.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Malware is defined as software designed to infiltrate or damage a computer system without the owner’s informed consent. Malware is actually a generic definition for all kind of computer threats. A simple classification of malware consists of file infectors and stand-alone malware. Another way of classifying malware is based on their particular action: worms, backdoors, trojans, rootkits, spyware, adware etc. Malware detection through standard, signature-based methods is getting more and more difficult since all current malware applications tend to have multiple polymorphic layers to avoid detection or to use side mechanisms to automatically update themselves to a newer version at short periods of time in order to avoid detection by any antivirus software.

1.2 OBJECTIVE

The growth in technology has not only provided opportunities to various types of business but also many cybercrimes. Introduction of technologies likeAI and machine learning has led to creation of intelligent and destructive malwares. Since the process is automated the number of these software’s areincreasing exponentially and a simple antivirus cannot keep up with that. Therefore, in order to face the challenges, we can use machine learning classifiers algorithms. We would use different supervised learning modelsto identify the pattern in malware files and then classify them as malicious and legitimate files.

1.3 INTENDED AUDIENCE

- Typically, such as an application service provider.
- B2C Providers/Organizations.
- Stock Analyst.
- Server monitoring manager.
- SOC Analyst.

1.4 SCOPE OF THE PROJECT

This model enables identification of malicious and unwanted software by multiple detection engines Respectively. we present a framework for malware detection aiming to get as few false positives as possible, by using a simple and a simple multi-stage combination (cascade) of different versions of the algorithm. Other automate classification algorithms could also be used in this framework, but we do not explore here this alternative. The *main steps* performed through this framework are A set of features is computed for every binary file in the training or test datasets, based on many possible ways of analyzing a malware and A machine learning model. This approach provides several important benefits including better detection of malicious Files, enhanced forensics capabilities and improved deploy ability. Access to a larger threat database without having to house it on your hard drive. Smaller installation agent for your antivirus software, so it takes up less space.Near real-time definition, updates based on data gathered from the entire network of users.

CHAPTER 2

PROBLEM DEFINITION

PROBLEM STATEMENT

As the requirement of cloud platform is increasing day by day, the attack on the cloud platform is also increasing. Malicious files or Malwares are the software, which are programmed to harm or create issue for the user or the system. Therefore, the files downloaded from the Internet (PE files) are under constant attack by cyber-criminals since they are typically low secured. There are various anti-malware companies that develop software to tackle malwares for e.g. Norton, McAfee etc. But even after the presence of all this software, malware attack still happens in a large number these techniques are not so efficient all the time or may fail if the file size is large or directly delete if the file is executable file.

.

CHAPTER 3

LITERATURE REVIEW

LITERATURE SURVEY

Sl.no	Author's Name/ Paper Title	Conference / year	Technology used	Results shared by author	What we infer ?
1.	Dr. Sanjay Silakari A Survey over the various malware detection techniques used incloud Computing.	IJERT 2019	Virtualization	Services provided by cloudand a Comparison of various techniques which used for the malware detection in cloud computing	A brief description over the techniques which used for malware detection in cloud computing
2.	OMER ASLAN A Comprehensive Review on Malware Detection Approaches.	IEEE 2020	Signature , Behaviour, heuristic based,Deep Learning, Cloud- based.	Effective and enhanced method to detect malware.	The research paper shows that to build an effective method to detect malware is a very challenging task, and there is a huge gap for new studies and methods.
3.	Babak BashariRad An Introduction to Docker and Analysis of its Performance	From Research Gate 2020	Docker performance andanalysis	Docker automates the applications when they are containerized. An extra layer of docker engine is added to the host operatingsystem	This research explains about the Docker platform in which the Cloud enables us to deploy applications into the clusters, monitor and manage them and also it helps in deploying the individual services froma single Docker image or deploy a multi- container stack.

4.	A Review of Cloud-Based Malware Detection System. Deepti Gupta	EJERS 2021	Cloud computing (Aws, Azure).	An effective approach to detect malware is a very challenging task and cloud-based approach is still at anearly stage by applying different detection approaches	This research explains advantages and disadvantages of cloud environments in detecting malware and proposes a cloud-based malware detection framework, which uses a hybrid approach to detect malware.
5.	Manan Kalpesh Integrating Machine Learningin Malware Detection Manan Kalpesh	2021	Machine Learning, Cybersecurity	The algorithm used for training the data was Gradient Boosted classifier and random forest classifierwhich gives us an accuracyof 98.764% and 99.311% respectively	This paper provides a rundown on the study of Malwares and techniques which can beuseful to analyze malicious software, which work on cloud computing in giving results. Usually, the results from static and dynamic analysis are insufficient therefore it directs the security analysts to use machine learning, deep learning, and neural network techniques.

CHAPTER 4
PROJECT DESCRIPTION

4.1 PROPOSED DESIGN

The clean files in the training database are mainly system files (from different versions of operating systems) and executable and library files from different popular applications. We also use clean files that are packed or have the same form or the same geometrical similarities with malware files in order to better train and test the system. The malware files in the training dataset have been taken from the Virus Share collection. The test dataset contains malware files from the dataset collection and clean files from different operating systems (other files that the ones used in the first database). Large dataset was used for testing the scaling up capabilities of the used machine learning algorithms.

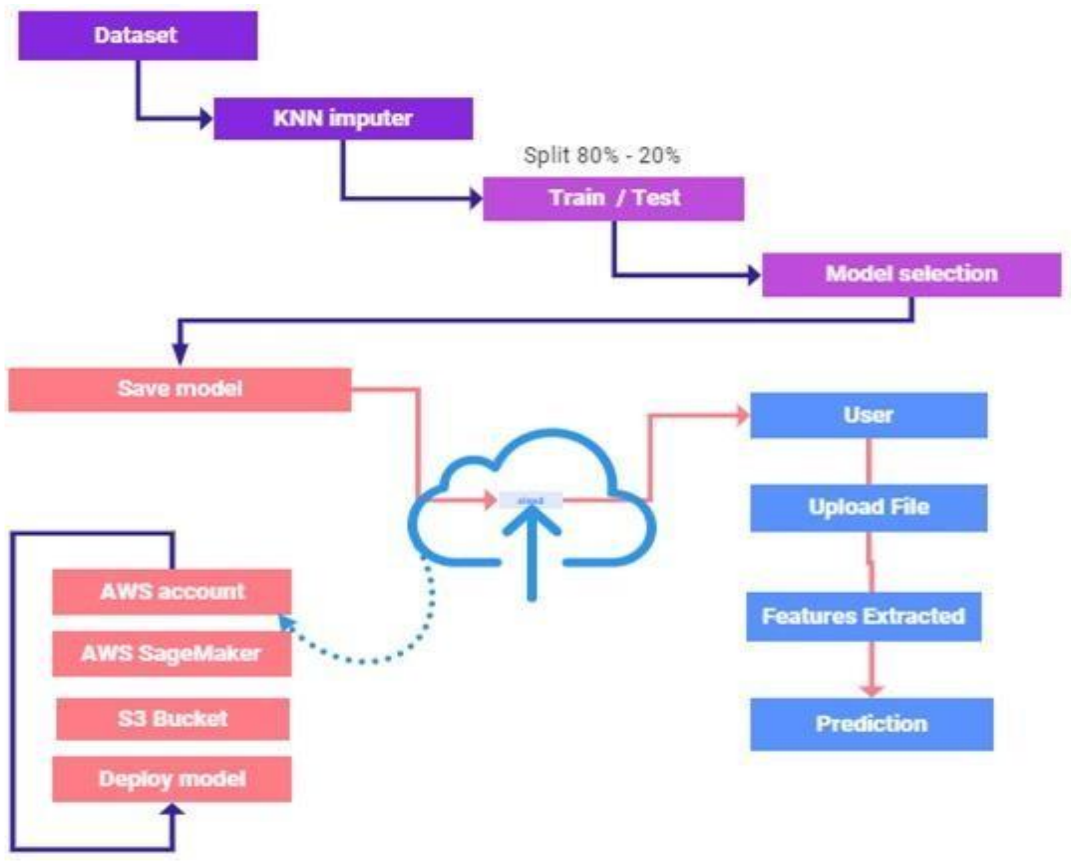


Fig.1. Design Diagram

1. Gathering Data:

The first real step of machine learning- Gathering Data. This step is very crucial as the quality and quantity of data gathered will directly determine how good the predictive model will turn out to be. The data collected is then tabulated and called as Training Data. The Virus-share Dataset.

2. KNN imputer:

The KNN imputer is used to fill out or predict the missing values in a dataset**.** It is a more useful method which works on the basic approach of the KNN algorithm rather than the naive approach of filling all the values with mean or the median.

3. Choosing a model:

The next step that follows in the workflow is choosing a model among the many that researchers and data scientists have created over the years. Make the choice of the right one that should get the job done.

4. Training:

The training process involves initializing some random values for say A and B of our model, predict the output with those values, then compare it with the model's prediction and then adjust the values so that they match the predictions that were made previously. This process then repeats and each cycle of updating is called one training step.

5. Evaluation:

Once training is complete, now we check if it is good enough using this step. Evaluation allows the testing of the model against data that has never been seen and used for training and is meant to be representative of how the model might perform when in the real world.

6. Prediction:

So this is the final step where you get to answer few questions. This is the point where the value of machine learning is realized. Here you can finally use your model to predict the outcome of what you want.

4.2 SCREEN DESIGN

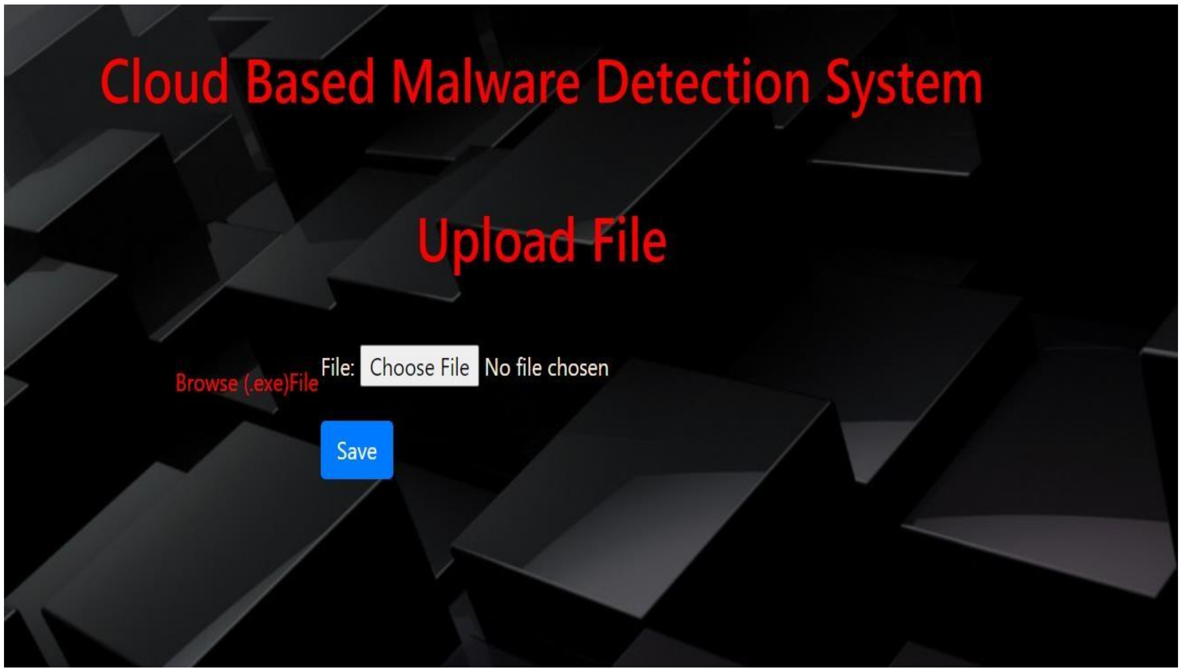


Fig.2. Screen Design

4.3 DATA FLOW DIAGRAM:

It represents the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system.

DFD COMPONENTS

DFD can represent Source, destination, storage and flow of data using the following set of components –

Entities - An external entity is a person, department, outside organization, or other information system that provides data to the system or receives outputs from the system.

Process - any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data Storage - files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”

Data Flow - the route that data takes between the external entities, processes and data stores

DFD-L0:

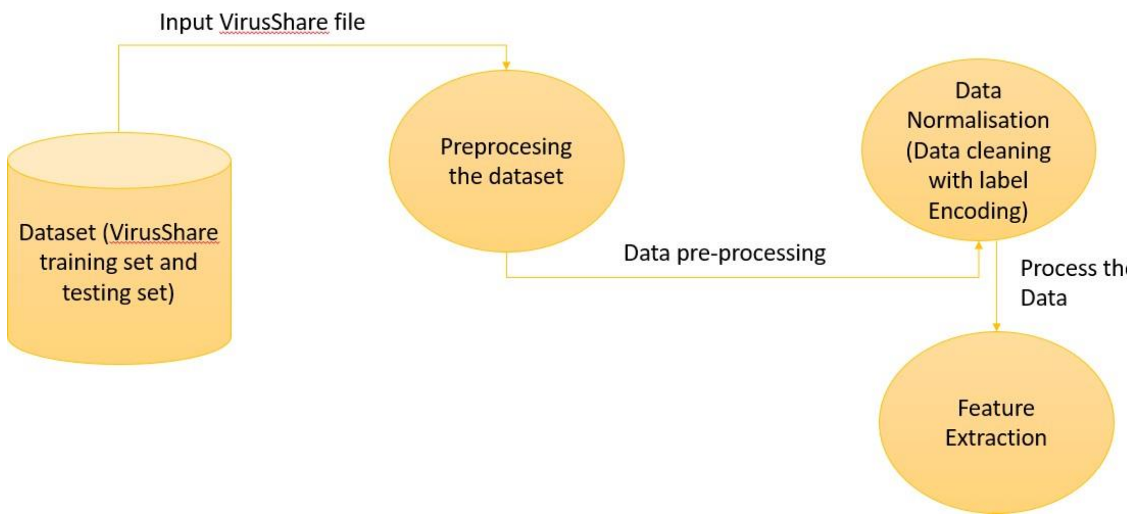

Input VirusShare file
Dataset (VirusShare training set and testing set)
Preprocesing the dataset
Data pre-processing
Data Normalisation (Data cleaning with label Encoding)
Process the Data
Feature Extraction

Fig.3. Data flow L0

DFD-L1:

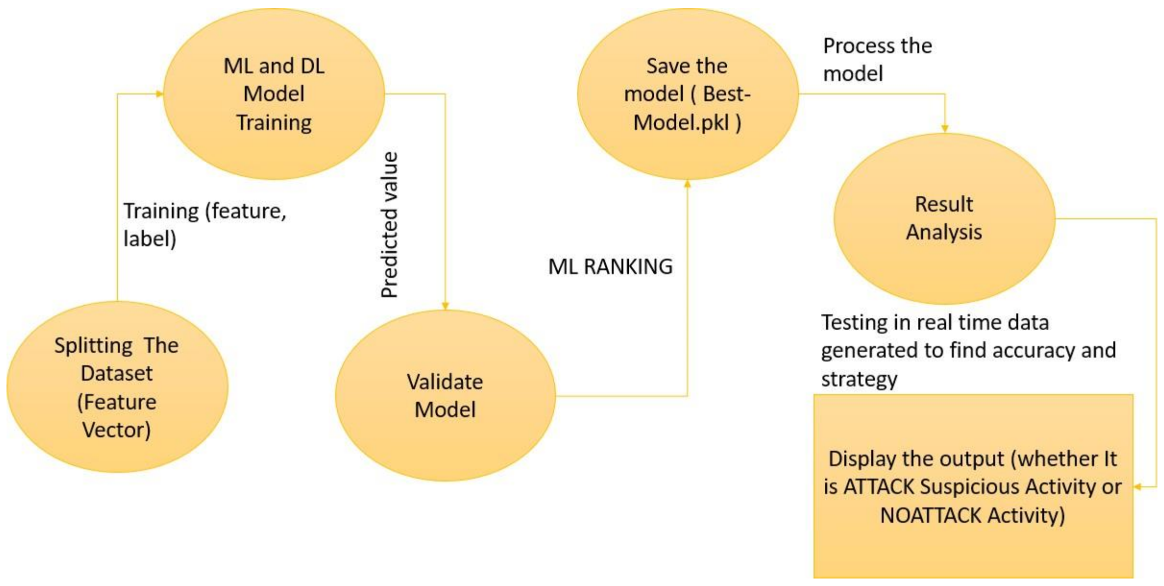

ML and DL Model Training
Training (feature, label)
Splitting The Dataset (Feature Vector)
Predicted value
Validate Model
ML RANKING
Save the model (Best-Model.pkl)
Process the model
Result Analysis
Testing in real time data generated to find accuracy and strategy
Display the output (whether It is ATTACK Suspicious Activity or NOATTACK Activity)

Fig.4. Data flow L1

4.4 USE CASE DIAGRAM:

Use case diagrams are used to gather the requirements of a system including internal and external influences. The purposes of use case diagrams is Used to get an outside view of a system and show the interaction among the requirements are actors.

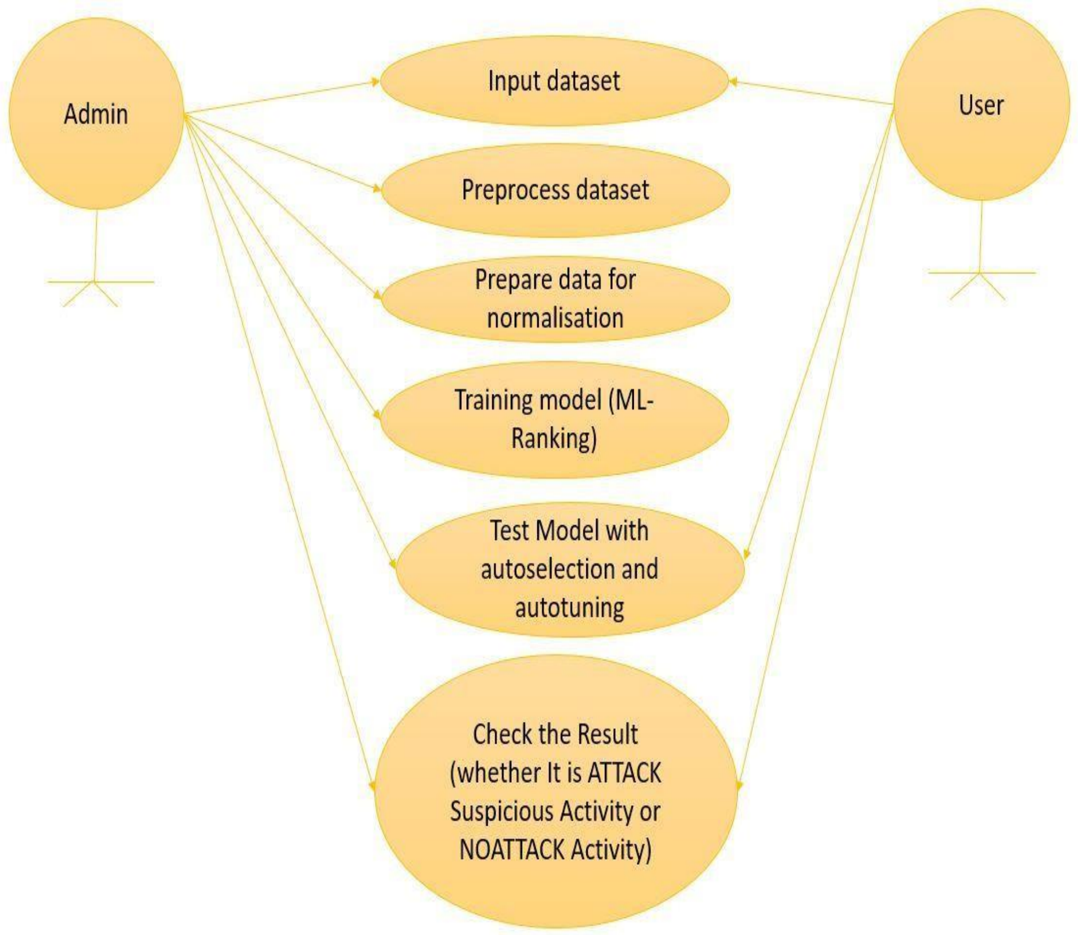

Admin
User
Input dataset
Preprocess dataset
Prepare data for normalisation
Training model (ML-Ranking)
Test Model with autoselection and autotuning
Check the Result (whether It is ATTACK Suspicious Activity or NOATTACK Activity)

Fig.5. Use Case Diagram

4.5 SEQUENCE DIAGRAM:

A sequence diagram is a system is an interaction diagram that shows how process operates with one and other and in what order. It depicts the objects and classes involved in the scenario and sequence of messages exchange between the objects needed to carry out the functionality of the scenario.

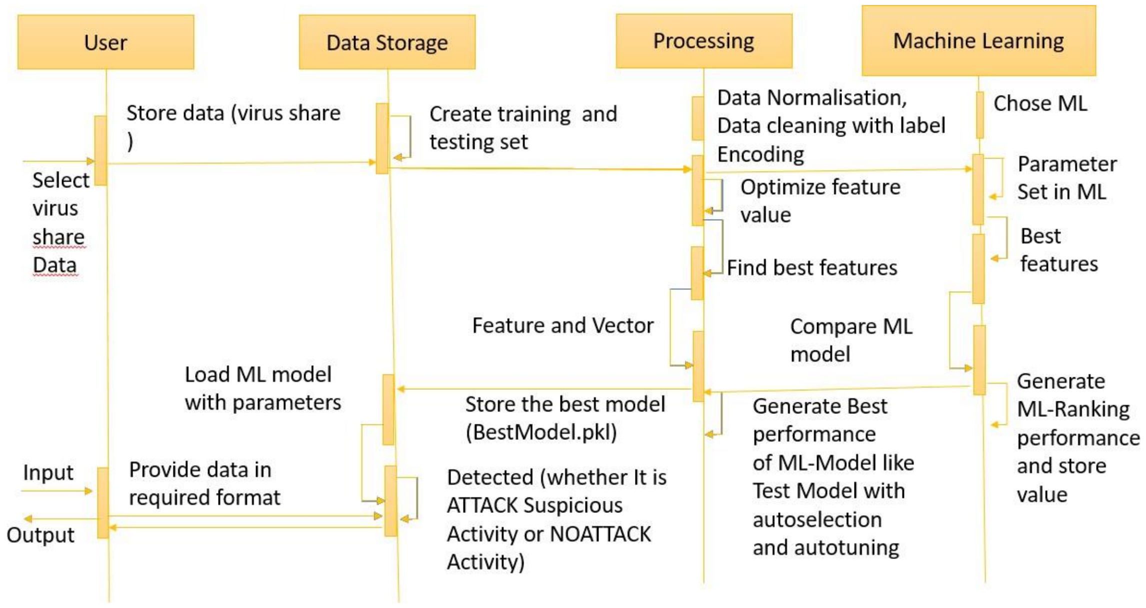

User
Data Storage
Processing
Machine Learning
Store data (virus share)
Create training and testing set
Data Normalisation, Data cleaning with label Encoding
Chose ML
Select virus share Data
Optimize feature value
Parameter Set in ML
Best features
Find best features
Feature and Vector
Compare ML model
Load ML model with parameters
Store the best model (BestModel.pkl)
Generate Best performance of ML-Model like Test Model with autoselection and autotuning
Generate ML-Ranking performance and store value
Input
Provide data in required format
Output
Detected (whether It is ATTACK Suspicious Activity or NOATTACK Activity)

Fig.6. Sequence Diagram

4.6 ASSUMPTIONS AND DEPENDENCIES

- To determine whether the files are malware or not, the Dataset should be validated.
- We are completely reliant on the cloud services.
- We are completely reliant on network resources.
- The Python Platform will be used for Machine Learning Algorithms.

CHAPTER 5
REQUIREMENTS

5.1 SOFTWARE & HARDWARE REQUIREMENTS

Hardware requirements

System: (Minimum) Intel i3 2.1 GHZ

Memory: 4GB.

Hard Disk: 80 GB.

Software requirements

Operating System: Windows 7 / 8/10.

Language: Python (3.7.4)

Tool: Anaconda Navigator, Jupyter Notebook.

Package: Scikit-Learn, Joblib.

5.2 FUNCTIONAL REQUIREMENTS

The functional requirements here are:

- Downloading Dataset from VirusShare containing malware and legitimate PE Files.
- Performing Malware Analysis and Classification, choosing best out of the used Machine Learning Algorithms and Feature classification.
- Deploying the model on the cloud for users to access the model for prediction from any part of the world.

5.3 NON-FUNCTIONAL REQUIREMENTS

- Usability
- Reliability
- Security
- Performance
- Portability
- Reusability

CHAPTER 6

METHODOLOGY

6.1. The Data Set:

- The dataset for the project is obtained from VirusShare.com is a repositoryof malware samples.
- These resources are open-source.

6.2. Feature Extraction:

- For developing model to detect malware a set of 11 PE features are used.These features are been extracted from a set of 57 PE features.
- A set of algorithms are run on the dataset of 57 PE features to get 11 bestfeatures of them.

6.3. PE File:

- PE stands for Portable Executable.
- PE is a format used by files this kind of file format is used by windowsapplications to run applications consisting of wrapped executable codes.

6.4.The Algorithms used:

1. DecisionTree:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

2. RandomForest:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

3. GradientBoosting:

Train the base model using the weighted training data

Then, add weak learners sequentially to make it a strong learner

Each weak learner consists of a decision tree; analyze the output of each decision tree and assign higher weights to the misclassified results. This gives more significance to the prediction with higher weights.

Continue the process until the model becomes capable of predicting the accurate result.

4. AdaBoost:

- Initially, Adaboost selects a training subset randomly.
- It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
- It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
- Also, It assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
- This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.
- To classify, perform a "vote" across all of the learning algorithms you built.

5. GNB:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge.
- It depends on the conditional probability.
- The formula for Bayes' theorem is given as:
- $P(A|B)= \frac{P(B|A)P(A)}{P(B)}$
- Where,
- P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

- P(B|A) is Likelihood probability: Probability of the evidence given that theprobability of a hypothesis is true.
- P(A) is Prior Probability: Probability of hypothesis before observing theevidence.
- P(B) is Marginal Probability: Probability of Evidence.

6.5. Machine Learning:

- Machine learning automatically learns from its experience and surroundings.
- The ML model uses algorithm that learns from experience or available data to predict the outcome.
- First a dataset is given as input to machine learning algorithm, the dataset consists of features & the machine learning analyses those features and predicts what the result can be.

6.5.1. Supervised Machine Learning:

- The supervised learning consists of both input and output data as dataset.
- The goal of the algorithm is to understand the mapping between the input and the output.

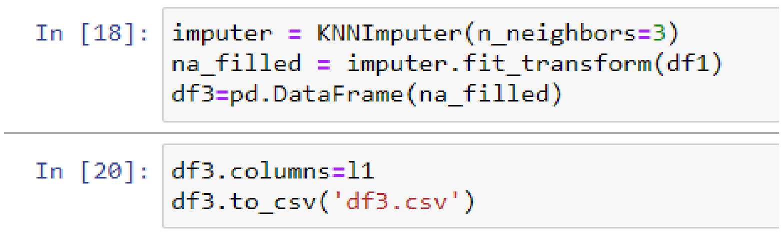
6.5.2. Un-Supervised Machine Learning:

- The Un-supervised machine learning is a type of learning in which only input data is given to the algorithm and the output is unknown.
- The algorithm has no supervision regarding the steps it should take. The algorithm has to analyses some pre-existing hidden pattern in the input dataset and make predictions or find a solution.

CHAPTER 7
EXPERIMENTATION

7.1 Description of code:

1. KNNimputer is a scikit-learn class used to fill out or predict the missing values in a dataset. It is a more useful method which works on the basic approach of the KNN algorithm rather than the naive approach of filling all the values with mean or the median. In this approach, we specify a distance from the missing values which is also known as the K parameter. The missing value will be predicted in reference to the mean of the neighbors.

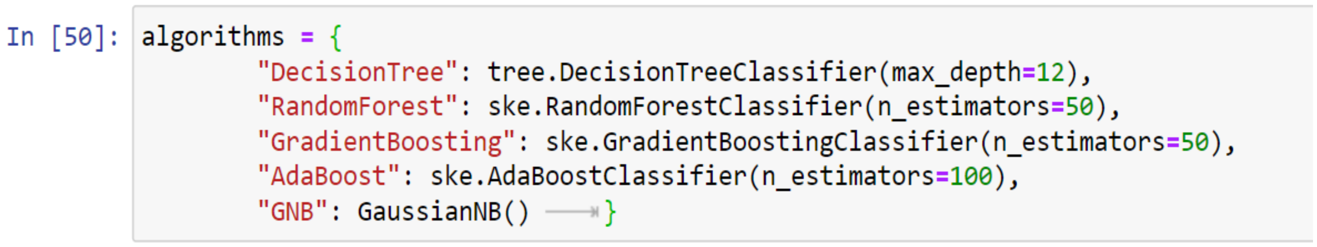


```
In [18]: imputer = KNNImputer(n_neighbors=3)
         na_filled = imputer.fit_transform(df1)
         df3=pd.DataFrame(na_filled)

In [20]: df3.columns=l1
         df3.to_csv('df3.csv')
```

Fig.7. KNN imputer

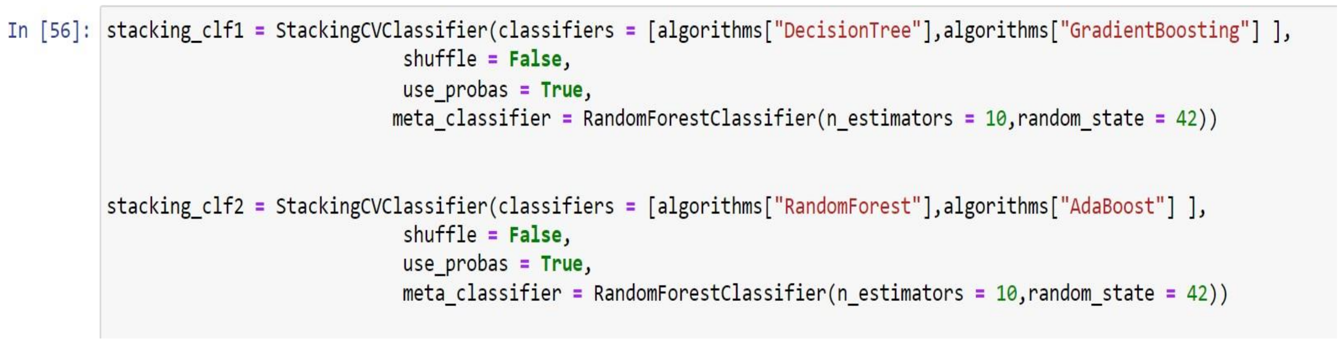
The algorithm above are used to specify the accuracy of the model where in the Decision tree. The max_depth parameter denotes maximum depth of the tree. In case, of random forest, these ensemble classifiers are the randomly created decision trees. Each decision tree is a single classifier and the target prediction is based on the majority voting method. n_estimators are the number of trees in the forest. In case, of boosting algorithms which means that they convert a set of weak learners into a single strong learner. They both initialize a strong learner (usually a decision tree) and iteratively create a weak learner that is added to the strong learner. They differ on how they create the weak learners during the iterative process. Finally theBayes theorem is based on conditional probability. The conditional probability helps us calculating the probability that something will happen.



```
In [50]: algorithms = {
                 "DecisionTree": tree.DecisionTreeClassifier(max_depth=12),
                 "RandomForest": ske.RandomForestClassifier(n_estimators=50),
                 "GradientBoosting": ske.GradientBoostingClassifier(n_estimators=50),
                 "AdaBoost": ske.AdaBoostClassifier(n_estimators=100),
                 "GNB": GaussianNB() ⟶}
```

Fig.8. Algorithms

Stacking is an ensemble machine learning algorithm that learns how to best combine the predictions from multiple well-performing machine learning models.



```
In [56]: stacking_clf1 = StackingCVClassifier(classifiers = [algorithms["DecisionTree"],algorithms["GradientBoosting"] ],
                                    shuffle = False,
                                    use_probas = True,
                                  meta_classifier = RandomForestClassifier(n_estimators = 10,random_state = 42))


         stacking_clf2 = StackingCVClassifier(classifiers = [algorithms["RandomForest"],algorithms["AdaBoost"] ],
                                    shuffle = False,
                                    use_probas = True,
                                    meta_classifier = RandomForestClassifier(n_estimators = 10,random_state = 42))
```

Fig.9. Stacking concept

CHAPTER 8
TESTING AND RESULTS

8.1 TEST CASE:

Table 1 Unit Test Case 1

S1 # Test Case	UTC- 1
Name of Test	Dataset(virus share testing-set, training set)
Expected Result	In this step, we aim to build through net to automatically download the targeted load virus share training and testing data from the Internet and store in dataset (each attack category) is used for evaluation of network anomaly and intrusion detection systems.
Actual output	Same as expected.
Remarks	Successful

Table 2 Unit Test Case 2

S2 # Test Case	UTC- 2
Name of Test	Pre-processing
Expected Result	The missing data of the dataset should be filled
Actual output	Same as expected.
Remarks	Successful

Table 3 Unit Test Case 3

S3 # Test Case	UTC- 3
Name of Test	Feature Extraction
Expected Result	From the pre-processed data important features should be extracted
Actual output	Same as expected.
Remarks	Successful

Table 4 Unit Test Case 4

S4 # Test Case	UTC- 4
Name of Test	Classification
Expected Result	The extracted features should be trained using the ML algorithms
Actual output	Same as expected.
Remarks	Successful

Table 5 Unit Test Case 5

S5 # Test Case	UTC- 5
Name of Test	Prediction
Expected Result	The uploaded exe file should be processed predict if the file is malware or not
Actual output	Same as expected.
Remarks	Successful

8.2 SNAPSHOTS:

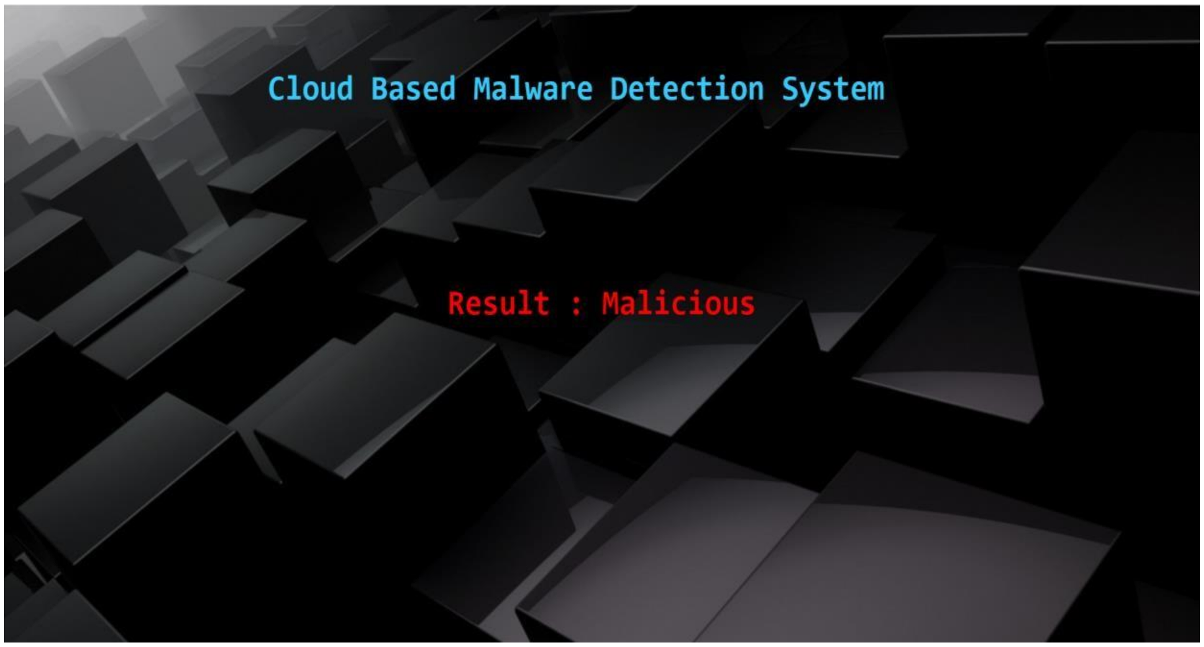


Fig.10. Result of Malware file

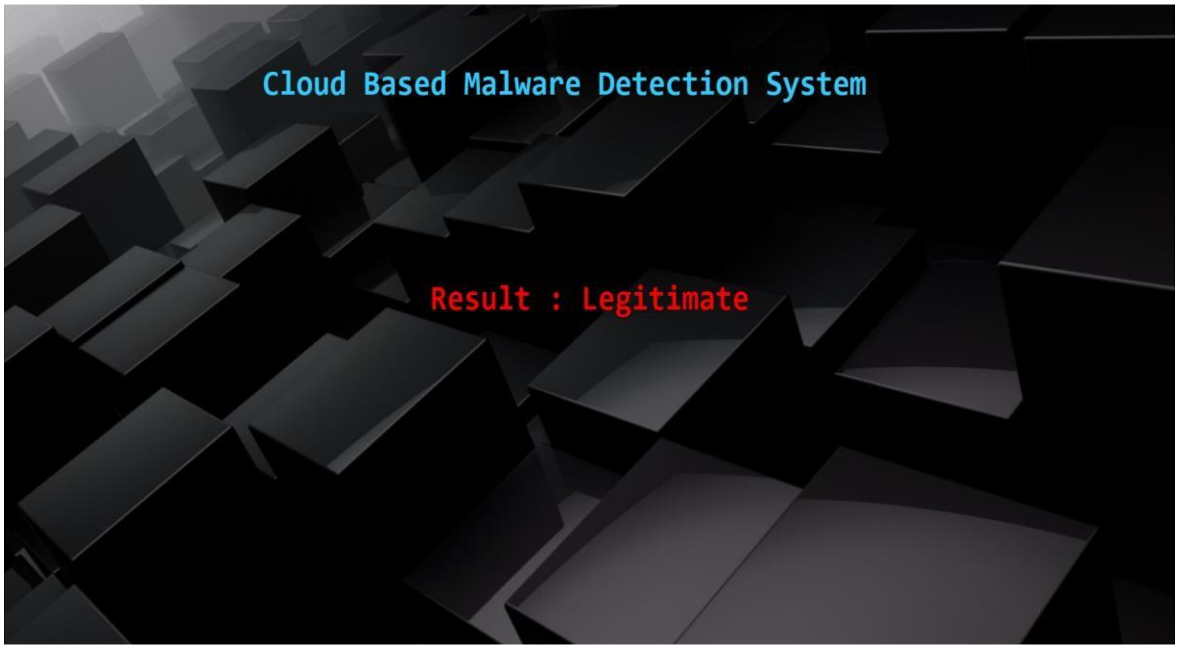


Fig.11. Result of Legitimate file

CHAPTER 9
CONCLUSIONS AND FUTURE WORK

To conclude, we have proposed a system for combined malware detection systems and cloud computing environments, all running binaries and malware are intercepted by submitting to one or more analysis engines, a complete check against a signature database to detect yet unknown exploits or malware. We will suggest increasing in the dependence of cloud computing as consumers increasingly move to cloud computing platforms for their computing needs. In this project, we reviewed previous work on malware detection, both conventional and in the presence of storage in order to determine the best approach for detection in the cloud. We also argue the benefits of distributing detection throughout the cloud and present a new approach to coordinate detection across the cloud.

In the proposed system, we have used traditional detection techniques (optimizing pattern) as per static signatures and dynamic detection technology (heuristic). Then, we have chosen for safer system methods as well as speed and modern to rival existing anti-virus.

REFERENCES

[1] Microsoft, “Microsoft security intelligence report”, [online]:http://www.microsoft.com/technet/security/default.mspx, July December 2006.

[2] Dropbox, Inc., dropbox.com webpage, [Online]: https://www.dropbox.Com/ (accessed 13/04/12).

[3] C. Grace. “Understanding intrusion-detection systems" [J], PC Network Advisor, vol. 122, pp. 11-15, 2000.

[4] S. Subashini, V. Kavitha s.l “A survey of security issues in service delivery models of cloud computing.” .Science Direct, Journal of Network and Computer Applications, pp. (1-11) January (2011).

[5] Shirlei Aparecida de Chaves, Rafael Brundo Uriarte and Carlos Becker Westphall “Toward an Architecture for Monitoring Private Clouds." S.l. IEEE December (2011).

[6] Bo Li, Eul Gyu I'm “ A signature matching optimization policy for anti-virus programs” Electronics and Computer Engineering, Hanyang University, Seoul, Korea. © IEEE 2011

[7] Chen, Z. & Yoon, J. “IT auditing to assure a secure cloud computing”, (2010). [Online]: http://doi.ieeecomputersociety.orgezproxy.umuc.edu/10.1109/SERVICES.2010.118.

[8] J. Oberheide, E. Cooke, and F. Jahanian “CloudAV: N-Version Antivirus in the Network Cloud”, In Proceedings of the 17th USENIX Security Symposium (Security'08). San Jose, CA, 2008.

[9] Jon Oberheide, Evan Cooke and Farnam Jahanian “Cloud N-Version Antivirus in the Network Cloud”,Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109 (2007).

[10] Matthias Schmidt, Lars Baumg Artner, Pablo Graubner, David Bock and Bernd Freisleben “Malware Detection and Kernel Rootkit Prevention in Cloud Computing Environments." University of Marburg, Germany (2011).

[11] K. Murad, S. Shirazi, Y. Zikria, and I. Nassar, “Evading Virus Detection Using Code Obfuscation” in Future Generation Information Technology, vol. 6485 of Lecture Notes in Computer Science, pp. 394–401, Springer Berlin , Heidelberg, 2010.

[12] Scott Treadwell, Mian Zhou “A Heuristic Approach for Detection of Obfuscated Malware”, Bank of America, 1201 Main St, Dallas, TX 75202, © IEEE 2009.

[13] Carlin, S., & Curran, K. “Cloud computing security”, International Journal of

Ambient Computing and Intelligence.

[14] “Heuristic analysis in Kaspersky Internet Security” [Online]: http://support.kaspersky.com , ID: 8936 , 2013 Mar 01 2013

[15] Algirdas Avizienis, “The n-version approach to fault-tolerant software”, IEEE Transactions on Software Engineering, 1985.
[16] Rodrigo Rodrigues, Miguel Castro, and Barbara Liskov. Base, “using abstraction to improve fault tolerance”, In Proceedings of the eighteenth ACM symposium on Operating systems principles, New York, NY, USA, 2001.

[17] Lajos Nagy, Richard Ford, and William Allen, “N-version programming for the detection of zero-day exploits”, In IEEE Topical Conference on Cybersecurity, Daytona Beach, Florida, USA, 2006.

[18] Carsten Willems and Thorsten Holz. Cwsandbox.[Online]: http://www.cwsandbox.org/, 2007.

[19] Hispasec Sistemas. “Virus total”, [Online]: http://virustotal.com, 2004.

[20] Norman Solutions. Norman sandbox whitepaper. http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf, 2003.

[21] Barracuda Networks. “Barracuda spam firewall”,[Online]: http://www.barracudanetworks.com, 2007.

[22] Cloudmark, “Cloudmark authority anti-virus”,[Online] :http://www.cloudmark.com, 2007.

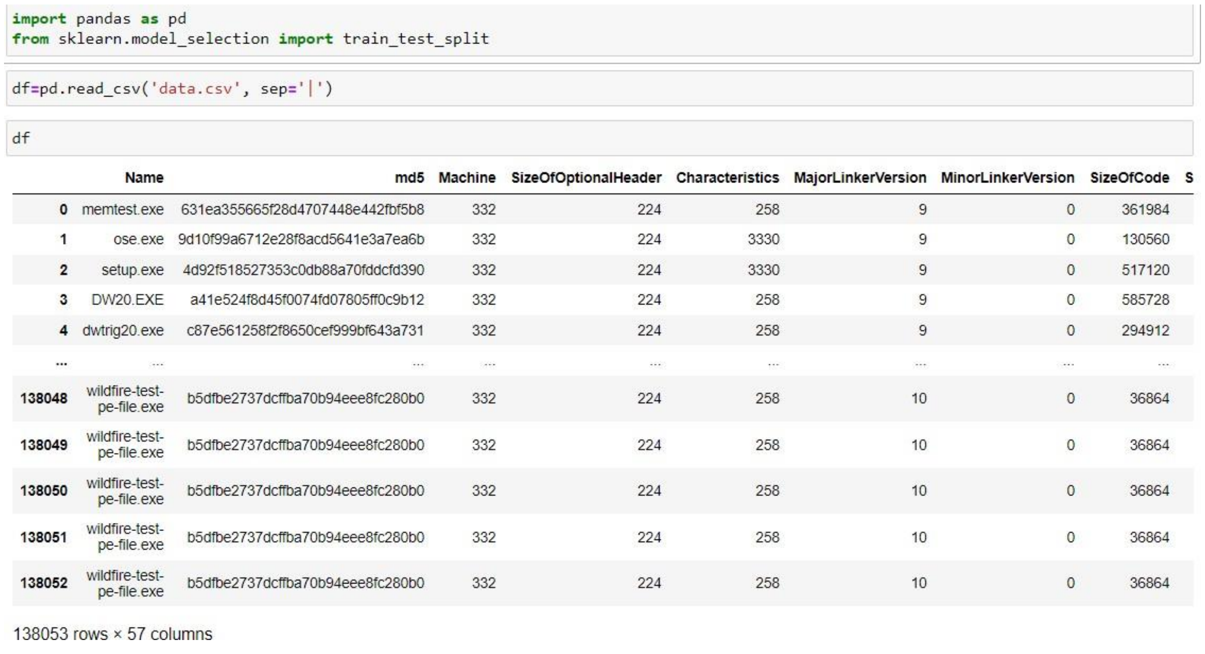
[23] Alexander Moshchuk, Tanya Bragin, Damien Deville, Steven D. Gribble, and Henry M. Levy, “Spyproxy: Execution-based detection of malicious web content”, In Proceedings of the 16th USENIX Security Symposium, August 2007.

[24] Stelios Sidiroglou, Angelos Stavrou, and Angelos D. Keromytis, “Mediated overlay services (moses): Network security as a composable service”, In Proceedings of the IEEE Sarnoff Symposium, Princeton, NJ, USA, 2007.

APPENDIX A

CODE

Defining file paths and Data Frame:



```
import pandas as pd
from sklearn.model_selection import train_test_split
```

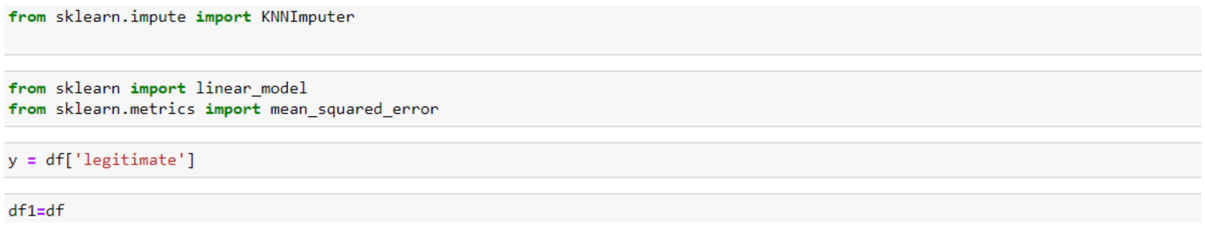
```
df=pd.read_csv('data.csv', sep='|')
```

```
df
```

```
                  Name                               md5  Machine  SizeOfOptionalHeader  Characteristics  MajorLinkerVersion  MinorLinkerVersion  SizeOfCode  S
     0     memtest.exe  631ea355665f28d4707448e442fbf5b8      332                   224              258                   9                   0      361984
     1         ose.exe  9d10f99a6712e28f8acd5641e3a7ea6b      332                   224             3330                   9                   0      130560
     2       setup.exe  4d92f518527353c0db88a70fddcfd390      332                   224             3330                   9                   0      517120
     3       DW20.EXE  a41e524f8d45f0074fd07805ff0c9b12      332                   224              258                   9                   0      585728
     4    dwtrig20.exe  c87e561258f2f8650cef999bf643a731      332                   224              258                   9                   0      294912
   ...             ...                               ...      ...                   ...              ...                 ...                 ...         ...
138048  wildfire-test-
           pe-file.exe  b5dfbe2737dcffba70b94eee8fc280b0      332                   224              258                  10                   0       36864
138049  wildfire-test-
           pe-file.exe  b5dfbe2737dcffba70b94eee8fc280b0      332                   224              258                  10                   0       36864
138050  wildfire-test-
           pe-file.exe  b5dfbe2737dcffba70b94eee8fc280b0      332                   224              258                  10                   0       36864
138051  wildfire-test-
           pe-file.exe  b5dfbe2737dcffba70b94eee8fc280b0      332                   224              258                  10                   0       36864
138052  wildfire-test-
           pe-file.exe  b5dfbe2737dcffba70b94eee8fc280b0      332                   224              258                  10                   0       36864

138053 rows × 57 columns
```

Defining KNN imputer for finding missing values:



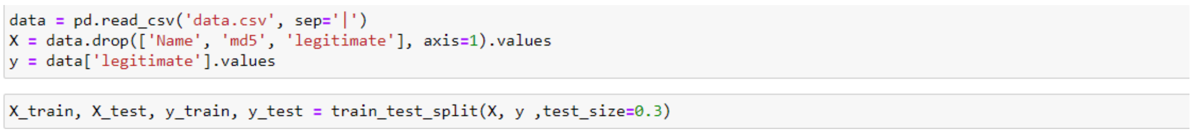
```
from sklearn.impute import KNNImputer
```

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
```

```
y = df['legitimate']
```

```
df1=df
```

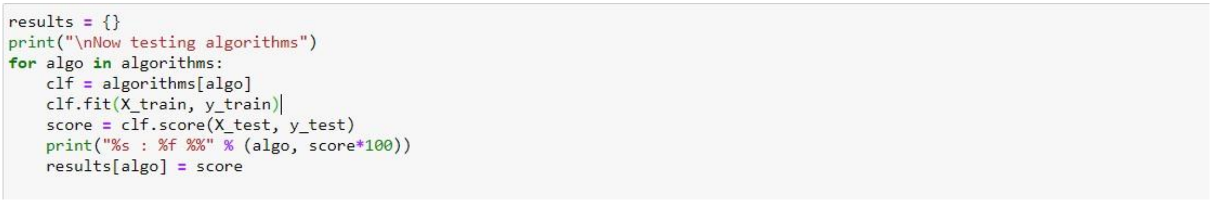
Defining Train, Test data:



```
data = pd.read_csv('data.csv', sep='|')
X = data.drop(['Name', 'md5', 'legitimate'], axis=1).values
y = data['legitimate'].values
```

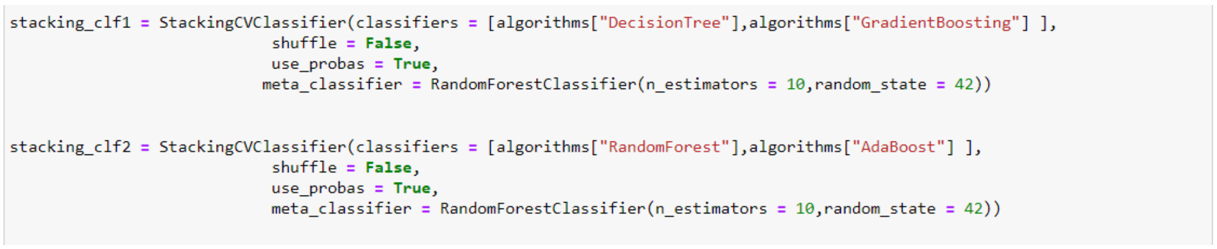
```
X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size=0.3)
```

Testing Accuracy of algorithm:



```
results = {}
print("\nNow testing algorithms")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    print("%s : %f %%" % (algo, score*100))
    results[algo] = score
```

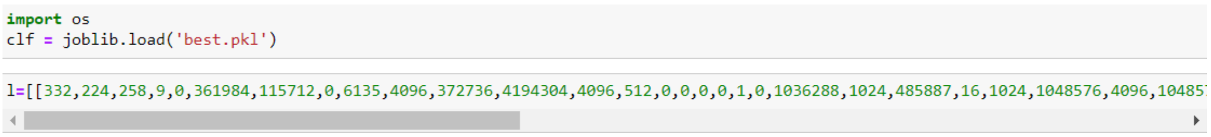
Defining Stacking concept:



```
stacking_clf1 = StackingCVClassifier(classifiers = [algorithms["DecisionTree"],algorithms["GradientBoosting"] ],
                            shuffle = False,
                            use_probas = True,
                           meta_classifier = RandomForestClassifier(n_estimators = 10,random_state = 42))


stacking_clf2 = StackingCVClassifier(classifiers = [algorithms["RandomForest"],algorithms["AdaBoost"] ],
                            shuffle = False,
                            use_probas = True,
                            meta_classifier = RandomForestClassifier(n_estimators = 10,random_state = 42))
```

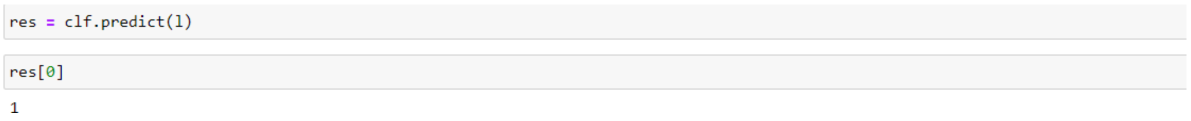
Uploading file:



```
import os
clf = joblib.load('best.pkl')
```

```
l=[[332,224,258,9,0,361984,115712,0,6135,4096,372736,4194304,4096,512,0,0,0,0,1,0,1036288,1024,485887,16,1024,1048576,4096,10485
```

Prediction:



```
res = clf.predict(l)
```

```
res[0]
```

```
1
```

Github Link:

https://github.com/zuhairBilal/Team34--CLOUD-BASED-MALWARE-DETECTION-SYSTEM.git

FUNDING AND PAPER PUBLICATION DETAILS

Paper Title	Cloud Based Malware Detection System
Journal	International Journal of Engineering Research & Technology(IJERT) Vol. 11 Issue 5, May-2022
Year of Publishing	2022
Abstract	Now a days the cloud environment has seen a sudden growth in its users and about 52% of the organisations have stepped up to use the cloud infrastructures just because of its flexible resources and economic scale it provides. When it comes to malware, malware is any type of software or a bug which tries to self-replicate or harm the hardware or a software of a system. These types of attacks are not known to the human eye as they are built with malicious intent to harm any system in use. So overall to overcome the problems faced and make a flexible solution, we propose a framework where Machine learning algorithms are used to find the best features from the data set provided by us and give an accuracy report, and the classifiers used among them best algorithm prediction will be used to extract the best features and use them. The .exe file features will be extracted and compared with the best algorithms extracted features to detect weather a give input file from a user is malware or a legitimate file. The use of the cloud infrastructure will be to deploy the said project so that it becomes convenient for a user to deploy the model.
Authors	Vinay J Zuhair Bilal Rohit K Syed Ali Zuhaib Dr Mouleeswaran SK

International Journal of Engineering Research & Technology

Volume - 11
Issue - 5
May 2022

IJERT

ISSN (Online) : 2278 - 0181

www.ijert.org
info@ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

Cloud based Malware Detection System

Vinay J, Zuhair Bilal, Rohit K, Syed Ali Zuhaib
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India

Dr. Mouleeswaran S K
Associate Professor
Department of Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India

Abstract— **Now a days the cloud environment has seen a sudden growth in its users and about 52% of the organisations have stepped up to use the cloud infrastructures just because of its flexible resources and economic scale it provides. When it comes to malware, malware is any type of software or a bug which tries to self-replicate or harm the hardware or a software of a system. These types of attacks are not known to the humaneye as they are built with malicious intent to harm any system in use. So overall to overcome the problems faced and make a flexible solution, we propose a frameworkwhere Machine learning algorithms are used to find the best features from the data set provided by us and give an accuracy report, and the classifiers used among them best algorithm prediction will be used to extract the best features and use them. The .exe file features will be extracted and compared with the best algorithms extractedfeatures to detect weather a give input file from a user is malware or a legitimate file. The use of the cloudinfrastructure will be to deploy the said project so that it becomes convenient for a user to deploy the model.**

Keywords—Machine Learning, Malware, detection, Legitimate file, Malware file, Cloud computing.

I. INTRODUCTION

Malware is defined as software designed to infiltrate or damage a computer system without the owner's informed consent. Malware is actually a generic definition for all kind of computer threats. A simple classification of malware consists of file infectors and stand-alone malware. Another way of classifying malware is based on their particular action: worms, backdoors, trojans, rootkits, spyware, adware etc. Malware detection through standard, signature based methods is getting more and more difficult since all current malware applications tend to have multiple polymorphic layers to avoiddetection or to use side mechanisms to automatically update themselves to a newer version at short periods of time in order to avoid detection by any antivirus software. For an example of dynamical file analysis for malware detection, via emulation in a virtual environment. Here we give a few references to exemplify such methods. Boosted decision trees working on n-grams are found to produce better results than both the Naive Bayes classifier and Support Vector Machines

Uses automatic extraction of association rules on Windows API execution sequences to distinguish between malware and clean program files. The *main steps* performed through this framework are sketched as follows:

1. A set of *features* is computed for every binary file in the training or test *datasets*, based on many possible ways of analyzing a malware.
2. A machine learning system based firstly on one-sided perceptrons, and then on feature mapped one-sided perceptrons and a kernelized one-sided perceptrons, combined with feature selection based on the F1 and F2 scores, is trainedon a medium-size dataset consisting of clean and malware files.

II. BACKGROUND

A. Cloud Computing

With the Internet's ubiquity in modern living, many argue that some level of cloud computing is now a common occurrence. This research heavily focuses on cloud computing technology, and thus requires a formal definition of cloud computing. Cloud computing cannot be easily defined. There are many definitions, which share the same common denominator: the Internet. Cloud computing is a way to use the Internet in the daily life of a single machine or single room, using all the tools installed on computers [Figure 1]. It is also the ability to use shared computing resources with local servers handling applications. With cloud computing users do not worry about the location and the storage of their data. They just start using the services anywhere and at any time. The main driver of this technology is Virtualization (Hypervisor) and virtual appliance.

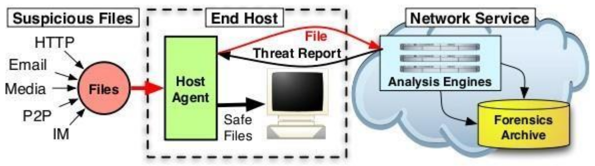


Fig. 1. The Flow of the Process of the cloud computing systems

Cloud computing offers different service models that allow customers to choose the appropriate service model that fits their environment needs, Cloud service models are software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a service (IaaS):

- Software-as-a-service (SaaS): The consumer uses the provider's applications, which are hosted in the cloud. For example, Salesforce.com CRM Application.
- Platform-as-a-service (PaaS): Consumers deploy their own applications into the cloud infrastructure. Programming languages and application development tools used must be supported by the provider. For example, Google Apps.
- Infrastructure-as-a-service (IaaS): Consumers are able to provide storage, network, processing, and other resources, and deploy and operate arbitrary software, ranging from applications to operating systems.

B. *Related Work*

As a matter of fact "cloud computing" concepts date backward to the 1950s, when large-scale mainframes were made available to schools and corporations, In addition, the on-demand computing concept of the cloud model went back to the time-sharing era in the 1960s. Therefore, many of the cloud computing security issues are arguably quite similar to the ones that were introduced during the Internet expansion era. However, Malware detection in a Cloud what we now commonly refer to as cloud computing is the result of an evolution of the widespread adoption of Virtualization,service-oriented architecture, autonomic, and utility computing. Details such as the location of infrastructure or component.

Devices are unknowns to most end-users, who no longer need to be thorough, understand or control the technology infrastructure that supports their computing activities. There are several previous studies related to this research dealing with all of cloud computing and its structure as well as detection systems used for each of the Static analysis, detection: Signature Optimizing Pattern Matching and Dynamic analysis detection: Heuristic.

C. *In Cloud Computing*

This novel paradigm provides significant advantages over traditional host-based an antivirus, including better detection of malicious software, enhanced forensic's capabilities, improved deployable and manageable retrospective detection. Use a production implementation and real-world deployment of the Cloud AV platform. An approach for combined malware detection and kernel rootkit prevention in virtualized cloud computing environments, and all running binaries in virtual instance are intercepted and submitted to one or more analysis engines. Besides a complete check against a signaturedatabase, lives introspection of all system calls is performed todetect yet unknown exploits or malware.

Malware detection has been an important issue in computing since the late '80s. Since then the predominant method of malware detection has been to scan a computer system for infection by matching malware signatures to files on the computer. Although detection of known samples is extremely reliable, signature based detection only works for malware that has been obtained, analyzed and a suitable signature identified. Decreased the signature mapping cost by optimizing signature library, taking advantage of common conduct characteristics of viruses such as self-replicate and seasoning, and proposed optimization policy against this scalability issue with the help of data mining. Moreover, he decreased the number of unnecessary signature matching and raises efficiency of that comparison procedure by rearrangement within a signature library. In Heuristic detection, Treadwell suggested analyzing the obfuscation pattern before unpacking, providing a chance to prevent malware from further execution.

III. SYSTEM DESIGN

The framework configuration prepare develops general structure building outline. Programming diagram incorporates addressing the item system works in a shape that might be changed into at least one anticipates. The essential demonstrated by the end customer must be placed in a systematical way. Diagram is a creative system; an extraordinary design is the best approach to reasonable structure. The structure "Layout" is portrayed as "The methodology of applying distinctive frameworks and guidelines with the ultimate objective of describing a strategy or a system in sufficient purpose important to permit its physical affirmation". Diverse design segments are taken after to add to the system. The design detail depicts the segments of the system, the sections or segments of the structure and their appearance to end-customers.

A. *Proposed System*

The clean files in the training database are mainly system files (from different versions of operating systems) and executable and library files from different popular applications. We also use clean files that are packed or have the same form or the same geometrical similarities with malware files (e.g use the same packer) in order to better train and test the system. The malware files in the training dataset have been taken from the Virus Share collection. The test dataset contains malware files from the dataset collection and clean files from different operating systems (other files that the ones used in the first database). Large dataset was used for testing the scaling up capabilities of the used machine learning algorithms.

B. *System Architecture*

The architectural configuration procedure is concerned with building up a fundamental basic system for a framework. It includes recognizing the real parts of the framework and interchanges between these segments. The beginning configuration procedure of recognizing these subsystems and building up a structure for subsystem control and correspondence is called construction modeling outline and the yield of this outline procedure is a portrayal of the product structural planning. The proposed architecture for this system is given below. It shows the way this system is designed and brief working of the system.

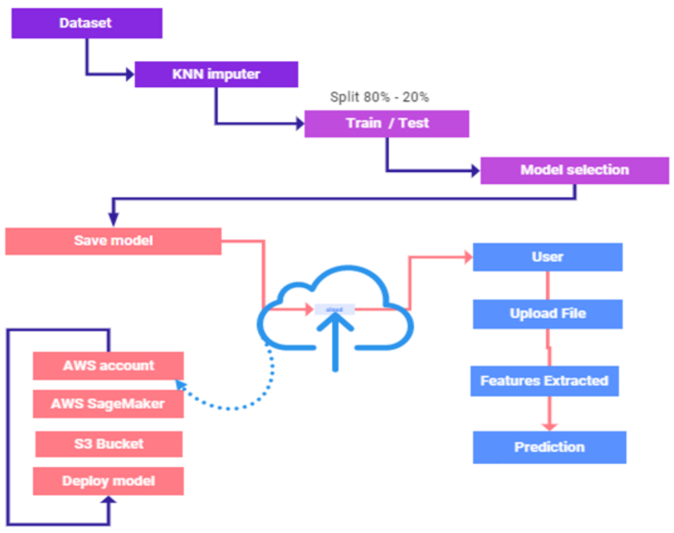


Fig. 2. System Architecture Model

C. *Data Flow Diagrams*

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements

DFD Components - DFD can represent Source, destination, storage and flow of data using the following set of components.

Entities - An external entity is a person, department, outside organization, or other information system that provides data to the system or receives outputs from the system.

Process - any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.

Data Storage - files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

Data Flow - the route that data takes between the external entities, processes and data stores.

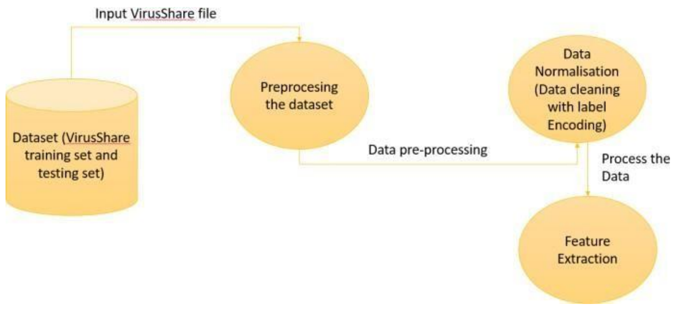

Input VirusShare file
Dataset (VirusShare training set and testing set)
Preprocesing the dataset
Data pre-processing
Data Normalisation (Data cleaning with label Encoding)
Process the Data
Feature Extraction

Fig. 3. Data Flow Diagram – L0

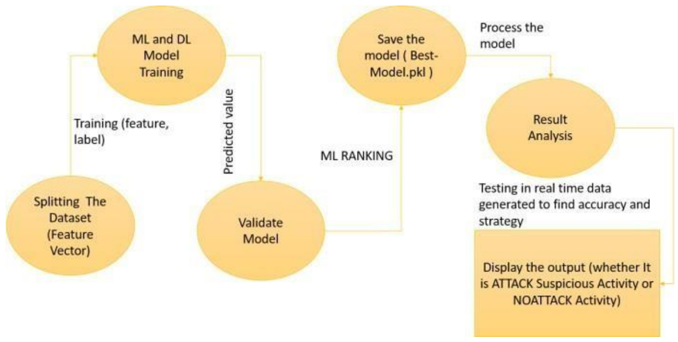

ML and DL Model Training
Save the model (Best-Model.pkl)
Process the model
Training (feature, label)
Predicted value
ML RANKING
Result Analysis
Testing in real time data generated to find accuracy and strategy
Splitting The Dataset (Feature Vector)
Validate Model
Display the output (whether It is ATTACK Suspicious Activity or NOATTACK Activity)

Fig. 4. Data Flow Diagram- L1

D. *Use Case Diagrams*

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and State chart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

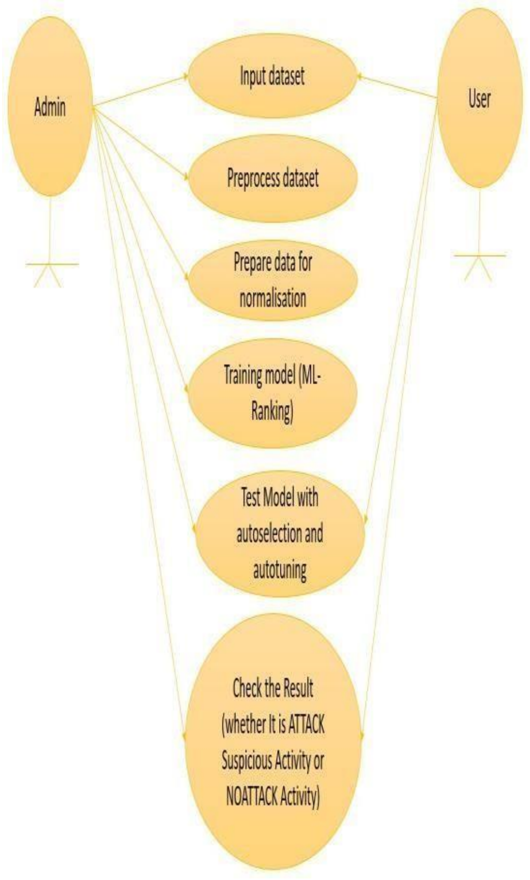

Admin
User
Input dataset
Preprocess dataset
Prepare data for normalisation
Training model (ML-Ranking)
Test Model with autoselection and autotuning
Check the Result (whether It is ATTACK Suspicious Activity or NOATTACK Activity)

Fig. 5. Use case Diagram of the Model

E. *Sequence Diagram*

A sequence diagram is a system is an interaction diagram that shows how process operates with one and other and in what order. It's a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and sequence of messages exchange between the objects needed to carry out the functionality of the scenario. Sequence diagram are sometimes called event diagrams orevent scenarios.

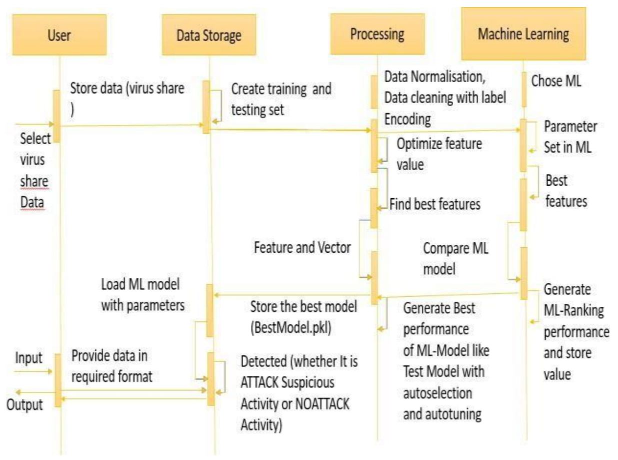

User
Data Storage
Processing
Machine Learning
Store data (virus share)
Create training and testing set
Data Normalisation, Data cleaning with label Encoding
Chose ML
Select virus share Data
Optimize feature value
Parameter Set in ML
Find best features
Best features
Feature and Vector
Compare ML model
Load ML model with parameters
Store the best model (BestModel.pkl)
Generate Best performance of ML-Model like Test Model with autoselection and autotuning
Generate ML-Ranking performance and store value
Input
Provide data in required format
Detected (whether It is ATTACK Suspicious Activity or NOATTACK Activity)
Output

Fig. 6. Sequence Diagram of the Model

IV. MODULES

This stage is the underlying stage in moving from issue to the course of action space. Accordingly, starting with what is obliged; diagram takes us to work towards how to full fill those requirements. System plot portrays all the critical data structure, record course of action, yield and genuine modulesin the structure and their Specification is picked. This assumes an essential part on the grounds that as it will give the last yield on which it was being working.

In our work we are using some modules, these modules are listed below.

A. *Gathering Data*

Once you know exactly what you want and the equipment's are in hand, it takes you to the first real step of machine learning- Gathering Data. This step is very crucial as the quality and quantity of data gathered will directly determine how good the predictive model will turn out to be. The data collected is then tabulated and called as Training Data.

1. THE VIRUS-SHARE DATASET

The VirusShare dataset is a repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code.

B. *Data Preparation*

After the training data is gathered, you move on to the next step of machine learning: Data preparation, where the data is loaded into a suitable place and then prepared for use in machine learning training. Here, the data is first put all together and then the order is randomized as the order of data should not affect what is learned.

This is also a good enough time to do any visualizations of the data, as that will help you see if there are any relevant relationships between the different variables, how you cantake their advantage and as well as show you if there are any data imbalances present. Also, the data now has to be split intotwo parts. The first part that is used in training our model, will be the majority of the dataset and the second will be used for the evaluation of the trained model's performance. The other forms of adjusting and manipulation like normalization, error correction, and more take place at this step.

C. *Choosing a Model*

The next step that follows in the workflow is choosing a model among the many that researchers and data scientists have created over the years. Make the choice of the right one that should get the job done.

D. *Training*

After the before steps are completed, you then move onto what is often considered the bulk of machine learning called training where the data is used to incrementally improve the model's ability to predict.

The training process involves initializing some random values for say A and B of our model, predict the output with those values, then compare it with the model's prediction and then adjust the values so that they match the predictions that were made previously.

This process then repeats and each cycle of updating is called one training step.

E. Evaluation

Once training is complete, you now check if it is good enough using this step. This is where that dataset you set aside earlier comes into play. Evaluation allows the testing of the model against data that has never been seen and used fortraining and is meant to be representative of how the model might perform when in the real world.

F. Parameter Tuning

Once the evaluation is over, any further improvement in your training can be possible by tuning the parameters. There were a few parameters that were implicitly assumed when the training was done. Another parameter included is the learning rate that defines how far the line is shifted during each step, based on the information from the previous training step. These values all play a role in the accuracy of the training model, and how long the training will take.

For models that are more complex, initial conditions play a significant role in the determination of the outcome of training. Differences can be seen depending on whether a model starts off training with values initialized to zeroes versus some distribution of values, which then leads to the question of which distribution is to be used. Since there are many considerations at this phase of training, it’s important that you define what makes a model good. These parameters are referred to as Hyper parameters. The adjustment or tuning of these parameters depends on the dataset, model, and the training process. Once you are done with these parameters andare satisfied you can move on to the last step.

G. Prediction

Machine learning is basically using data to answer questions. So this is the final step where you get to answer few questions. This is the point where the value of machine learning is realized. Here you can finally use your model to predict the outcome of what you want.

The above-mentioned steps take you from where you create a model to where you predict its output and thus acts as a learning path.

V. METHODOLOGY

A. Random Forest Algorithm

Random forest is a type of supervised machine learning algorithm based on <u>ensemble learning</u>. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The <u>random forest</u> algorithm combines multiple algorithm of the same type i.e. multipledecision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.
The following are the **basic steps** involved in performing the random forest algorithm:

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

B. Gaussian Naïve Bayes

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.
An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co- variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

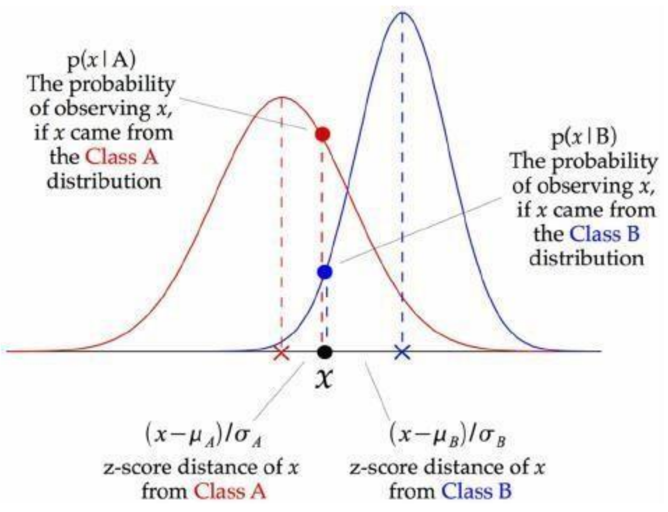

$p(x \mid A)$
The probability of observing x, if x came from the Class A distribution
$p(x \mid B)$
The probability of observing x, if x came from the Class B distribution
x
$(x-\mu_A)/\sigma_A$
z-score distance of x from Class A
$(x-\mu_B)/\sigma_B$
z-score distance of x from Class B

Fig. 7. The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works.

At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

The conditional probability of event A given event B means the probability of event A occurring given that event B has already occurred. Mathematically, the conditional probability of A given B can be denoted as P[A|B] = P[A AND B] / P[B].

C. *Decision Tree Algorithm*

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.
The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).
In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

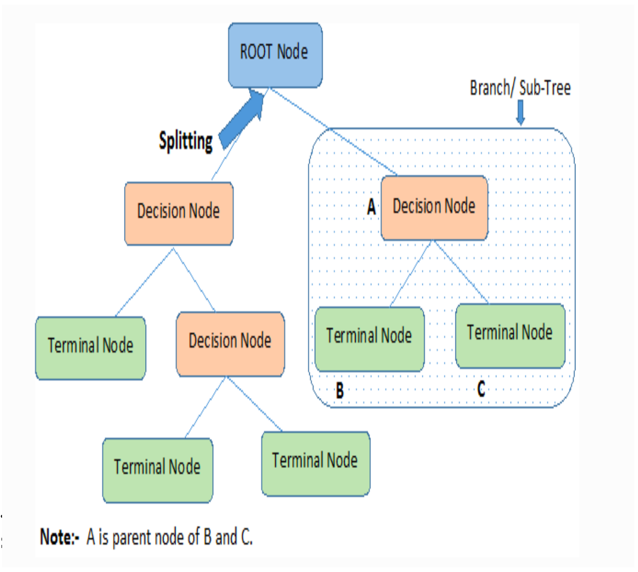


Fig.
Cla

1. WORKING PROCEDURE

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees.
Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split, which results in most homogeneous sub-nodes.

VI. CONCLUSION AND FUTURE WORK

To conclude, we have proposed a system for combined malware detection systems and cloud computing environments, all running binaries and malware areintercepted by submitting to one or more analysis engines, a complete check against a signature database to detect yet unknown exploits or malware. We will suggest increasing in the dependence of cloud computing as consumers increasinglymove to cloud computing platforms for their computing needs.In this paper, we reviewed previous work on malware detection, both conventional and in the presence of storage in order to determine the best approach for detection in the cloud. We also argue the benefits of distributing detection throughoutthe cloud and present a new approach to coordinate detection across the cloud.
In the proposed system, we have used traditional detection techniques (optimizing pattern) as per static signatures and dynamic detection technology (heuristic). Then, we have chosen for safer system methods as well as speed and modern to rival existing anti-virus.
The proposal of this work is to find the best solutions to the problems of anti-viruses and improve performance and find possible alternatives for a better working environment without problems with high efficiency and flexibility.
We used the optimal traditional methods and modern to detect viruses, for unknown and already detected viruses through the signatures and the Heuristic.
Future work in this field will focus on the development of detection systems based on memory introspection and heuristic or statistical detection, as opposed to signature-baseddetection.

ACKNOWLEDGMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work. First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.
We would like to thank **Dr. A Srinivas. Dean**, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice.It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman**, **Computer Science, and Engineering**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible. We would like to thank our guide **Dr Mouleeswaran SK, Associate Professor**, **Dept. of Computer Science and Engineering**, **Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and the fruitful culmination of the project.
We would like to thank our Project Coordinator **Dr. Meenakshi Malhotra** and all the staff members of Computer Science and Engineering for their support.
We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Project work.

REFERENCES

[1] Microsoft, "Microsoft security intelligence report", [online]:http://www.microsoft.com/technet/security/default.mspx, July December 2006.

[2] Dropbox, Inc., dropbox.com webpage, [Online]: https://www.dropbox.Com/ (accessed 13/04/12).

[3] C. Grace. "Understanding intrusion-detection systems" [J], PC Network Advisor, vol. 122, pp. 11-15, 2000.

[4] S. Subashini, V. Kavitha s.l "A survey of security issues in service delivery models of cloud computing." .Science Direct, Journal of Network and Computer Applications, pp. (1-11) January (2011).

[5] Shirlei Aparecida de Chaves, Rafael Brundo Uriarte and Carlos Becker Westphall "Toward an Architecture for Monitoring Private Clouds." S.l. IEEE December (2011).

[6] Bo Li, Eul Gyu I'm " A signature matching optimization policy for anti-virus programs" Electronics and Computer Engineering, Hanyang University, Seoul, Korea. © IEEE 2011

[7] Chen, Z. & Yoon, J. "IT auditing to assure a secure cloud computing", (2010). [Online]: http://doi.ieeecomputersociety.orgezproxy.umuc.edu/10.1109/SERVICES.2010.118.

[8] J. Oberheide, E. Cooke, and F. Jahanian "CloudAV: N- Version Antivirus in the Network Cloud", In Proceedings of the 17th USENIX Security Symposium (Security'08). San Jose, CA, 2008.

[9] Jon Oberheide, Evan Cooke and Farnam Jahanian "Cloud N-Version Antivirus in the Network Cloud",Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109 (2007).

[10] Matthias Schmidt, Lars Baumg Artner, Pablo Graubner, David Bock and Bernd Freisleben "Malware Detection and Kernel Rootkit Prevention in Cloud Computing Environments." University of Marburg, Germany (2011).

[11] K. Murad, S. Shirazi, Y. Zikria, and I. Nassar, "Evading Virus Detection Using Code Obfuscation" in Future Generation Information Technology, vol. 6485 of Lecture Notes in Computer Science, pp. 394–401, Springer Berlin , Heidelberg, 2010.

[12] Scott Treadwell, Mian Zhou "A Heuristic Approach for Detection of Obfuscated Malware", Bank of America, 1201 MainSt, Dallas, TX 75202, © IEEE 2009.

[13] Carlin, S., & Curran, K. "Cloud computing security", International Journal of Ambient Computing and Intelligence.

[14] "Heuristic analysis in Kaspersky Internet Security" [Online]: http://support.kaspersky.com , ID: 8936 , 2013 Mar 01 2013

[15] Algirdas Avizienis, "The n-version approach to fault- tolerant software", IEEE Transactions on Software Engineering, 1985.

[16] Rodrigo Rodrigues, Miguel Castro, and Barbara Liskov. Base, "using abstraction to improve fault tolerance", In Proceedings of the eighteenth ACM symposium on Operating systems principles, New York, NY, USA, 2001.

[17] Lajos Nagy, Richard Ford, and William Allen, "N- version programming for the detection of zero-day exploits", In IEEE Topical Conference on Cybersecurity, Daytona Beach, Florida, USA, 2006.

[18] Carsten Willems and Thorsten Holz. Cwsandbox.[Online]: http://www.cwsandbox.org/, 2007.

[19] Hispasec Sistemas. "Virus total", [Online]: http://virustotal.com, 2004.

[20] Norman Solutions. Norman sandbox whitepaper. http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf, 2003.

[21] Barracuda Networks. "Barracuda spam firewall",[Online]: http://www.barracudanetworks.com, 2007.

[22] Cloudmark, "Cloudmark authority anti-virus",[Online]:http://www.cloudmark.com, 2007.

[23] Alexander Moshchuk, Tanya Bragin, Damien Deville, Steven D. Gribble, and Henry M. Levy, "Spyproxy: Execution-based detection of malicious web content", In Proceedings of the 16th USENIX Security Symposium, August 2007.

[24] Stelios Sidiroglou, Angelos Stavrou, and Angelos D. Keromytis, "Mediated overlay services (moses): Network security as a composable service", In Proceedings of the IEEE Sarnoff Symposium, Princeton, NJ, USA, 2007.