UNVEILING MALWARE WITH AUTOENCODER BASED DETECTION

A PROJECT REPORT

Submitted by

PREETHIKA A.R **620320104055**

SNEHA P **620320104068**

VALLARASI P **620320104079**

RUBADHARSINI S **620320104701**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

BHARATHIYAR INSTITUTE OF ENGINEERING FOR WOMEN

DEVIYAKURICHI - 636112

ANNA UNIVERSITY, CHENNAI :: 600 025

MAY 2024

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**UNVEILING MALWARE WITH AUTOENCODER BASED DETECTION**" is the bonafide work of **PREETHIKA A R (620320104055), SNEHA P (620320104068), VALLARASI P (620320104079) RUBADHARSINI S (620320104701)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. P. MADHUBALA, M.E., Ph.D.,	**Dr. P. MADHUBALA, M.E.,Ph.D.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Associate Professor	**Associate Professor**
Department of Computer	Department of Computer
Science And Engineering	Science And Engineering
Bharathiyar Institute of	Bharathiyar Institute of
Engineering For Women	Engineering For Women
Deviyakurichi – 636112.	Deviyakurichi – 636112.

Submitted for the University Viva Voce Examination held on……

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

ACKNOWLEDGEMENT

This project work is dedicated to Almighty God for blessing us with inspirational parents, teachers and good friends. We like to thank our chairperson **Mrs.S.LEELAVATHI ELAYAPPAN** for her sincere endeavors to her premier institution.

We like to express our deep gratitude to beloved secretary **Dr.A.K.RAMASAMY**, for his enthusiastic motivation in computing this project and also, thanks to all our directions of Sri Sakthi Educational Trust.

We wish to express our profound thanks to our beloved principal **Dr.R.PUNIDHA,M.E.,Ph.D.,** for her encouragement during the project of study.

We feel elated to keep on record our heartful thanks and gratitude to our vice- principal **Mr.K.KALAISELVAN,M.E.,** for his timely help during period of the project.

We solemnly express our gratitude to our Head of the Department **Dr.P.MADHUBALA., M.E, Ph.D**., for her encouragement and constant support which initiated me to complete this project work.

We indebted to our guide **Dr.P.MADHUBALA., M.E, Ph.D**., for her constant guidance and encouragement throughout the report work without which the project could not been completed successfully.

Finally, we wish to thank all teaching and non- teaching staff members of Department of Computer Science and Engineering, for their support towards the successful completion of this technical project work.

ABSTRACT

The Proposed System introduces the versatile framework designed to effectively discern between malware files and clean files while prioritizing the reduction of false positives. Our approach utilizes a combination of machine learning algorithms, with a focus on cascade one-sided perceptrons and cascade kernelized one-sided perceptrons. Initially, we elucidate the conceptual underpinnings of our framework, highlighting the sequential learning mechanism of cascade perceptrons and the enhanced discriminatory capabilities afforded by kernelization. Through experimentation on medium-sized datasets encompassing diverse samples of malware and clean files, we demonstrate the efficacy of our approach in achieving high detection accuracy while minimizing false positives. Subsequently, recognizing the necessity to adapt our framework for handling very large datasets, we undertake a scaling-up process. This process involves optimizing computational resources, parallelizing computations, and implementing distributed learning strategies to facilitate the analysis of extensive datasets. By successfully navigating the scaling-up process, our framework is equipped to tackle the challenges posed by large-scale malware detection, thereby advancing cybersecurity defenses in an era of evolving threats.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The system software designed to infiltrate or damage a computer system without the owner's in- formed 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. For an example of dynamical file analysis for malware detection, via emulation in a virtual environment, the interested reader can see Classical methods for the detection of metamorphic viruses are described in . An overview on different machine learning methods that were proposed for malware detection is given in . Here we give a few references to exemplify such methods. - In boosted decision trees working on n-grams are found to produce better results than both the Naive Bayes clasifier and Support Vector Machines. uses automatic extraction of association rules on Windows API execution sequences to distinguish between malware and clean program files. Also using association rules, but on honeytokens of known parameters, is - In Hidden Markov Models are used to detect whether a given program file is (or is not) a variant of a previous program file. To reach a similar goal, employs Profile Hidden Markov Models, which have been previously used with great success for sequence analysis in bioinformatics. - The capacity of neural networks to detect polymorphic malware is explored in , Self-Organizing Maps .

CHAPTER 2

LITERATURE SURVEY

2.1 Title: Cloud security architecture based on user authentication and symmetric key cryptographic techniques, 2020

Author: Abdul Raoof

Technologies and Algorithm Used:

The study is implemented on the Structure for cloud security with efficient security in communication system and AES based file encryption system. This security architecture can be easily applied on PaaS, IaaS and SaaS and one time password provides extra security in the authenticating users.

Advantages:

Performance time and accuracy

Disadvantages:

Training model prediction on Time is High

It is based on Low Accuracy

2.2 Title: Analysis and Countermeasures for Security and Privacy Issues in Cloud Computing, 2019

Author: Q. P. Rana, Nitin Pandey

Technologies and Algorithm Used:

The cloud computing environment is adopted by a large number of organizations so the rapid transition toward the clouds has fuelled concerns about security perspective. There are numbers of risks and

challenges that have emerged due to use of cloud computing. The aim of this paper is to identify security issues in cloud computing which will be helpful to both cloud service providers and users to resolve those issues. As a result, this paper will access cloud security by recognizing security

Advantages:

More effective and efficient**.**

Disadvantages:

Not give accurate prediction result.

2.3 Title: Using Firefly and Genetic Metaheuristics for Anomaly Detection based on Network Flows, 2014

Author: Faisal Hussain

Technologies and Algorithm Used:

In this work, we proposed a Traffic monitoring is a challenging task which requires efficient ways to detect every deviation from the normal behavior on computer networks. In this paper, we present two models to detect network anomaly using flow data such as bits and packets per second based on: Firefly Algorithm and Genetic Algorithm. Both results were evaluated to measure their ability to detect network anomalies, and results were then compared. We experienced good results using data collected at the backbone of a university.

Advantages:

Efficiency measure and the accuracy

Disadvantages:

Not give accurate prediction result.

2.4 Title: A Multiple-Layer Representation Learning Model for Network-Based Attack Detection, 2018

Author: Suresh M

Technologies and Algorithm Used:

The proposed solutions are this ensures fine-grained detection of various attacks. The proposed framework has been compared with the existing deep learning models using three real datasets (a new dataset NBC, a combination of UNSW-NB15 and CICIDS2017 consisting of 101 classes).

Advantages:

It performs accurate classification of health state in comparison with other methods

Disadvantages:

It is low in efficiency.

2.5 Title: Detecting Distributed Denial of Service Attacks Using Data Mining Techniques, 2018

Author: Linga

Technologies and Algorithm Used:

In this study, we DDoS (Distributed Denial of Service) attack has affected many IoT networks in recent past that has resulted in huge losses. We have proposed deep learning models and evaluated those using latest CICIDS2017

datasets for DDoS attack detection which has provided highest accuracy as 97.16% also proposed models are compared with machine learning algorithms

Advantages:

The proposed solution can successfully detect network intrusions and DDOS communication with high precision**.**

More Reliable.

Disadvantages:

It is less in efficiency and not give perfect result.

This finding is disadvantageous to the organization experiencing such attack.

The difficulty in identifying all articles that are related to this study.

CHAPTER 3

PROBLEM ANALYSIS

3.1 EXISTING SYSTEM

In existing system, The malware files in the training dataset have been taken from the Virus Heaven collection. The test dataset contains malware files from the WildList collection and clean files from different operating systems (other files that the ones usedin the first database). The malware collection in the training and test datasets consists of trojans, backdoors, hacktools, rootkits, worms and other types of malware. The first and third columns in Table II represent the percentage of those malware types from the total number of files of the training and respectively test datasets. The second column in Table II represents the corresponding percentage of malware unique combinations from the total number of unique combinations of feature values for the training dataset

3.2 **DISADVANTAGES**

Doesn’t Efficient for handling large volume of data.

Theoretical Limits

Incorrect Classification Results.

Less Prediction Accuracy.

3.3 **PROPOSED SYSTEM**

The proposed model is introduced to overcome all the disadvantages that arises in the existing system. This system will increase the accuracy of the classification results by classifying the data based on the software quality prediction dataset and others using SVM , Gradient Boosting ,Navie Bayes Random forest and decision Tree algorithms.It enhances the performance of the overall classification results.

3.4 ADVANTAGES

High performance.

Provide accurate prediction results.

It avoid sparsity problems.

Reduces the information Loss and the bias of the inference due to the multiple estimates.

CHAPTER 4
SYSTEM SPECIFICATIONS

4.1 SOFTWARE REQUIREMENTS

O/S : Windows 7.

Language : Python

Front End : Anaconda Navigator - PYCHARM

4.2 HARDWARE REQUIREMENTS

System : Pentium IV 2.4 GHz

Hard Disk : 512 GB

Mouse : Logitech.

Keyboard : 110 keys enhanced

Ram : 4 GB

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

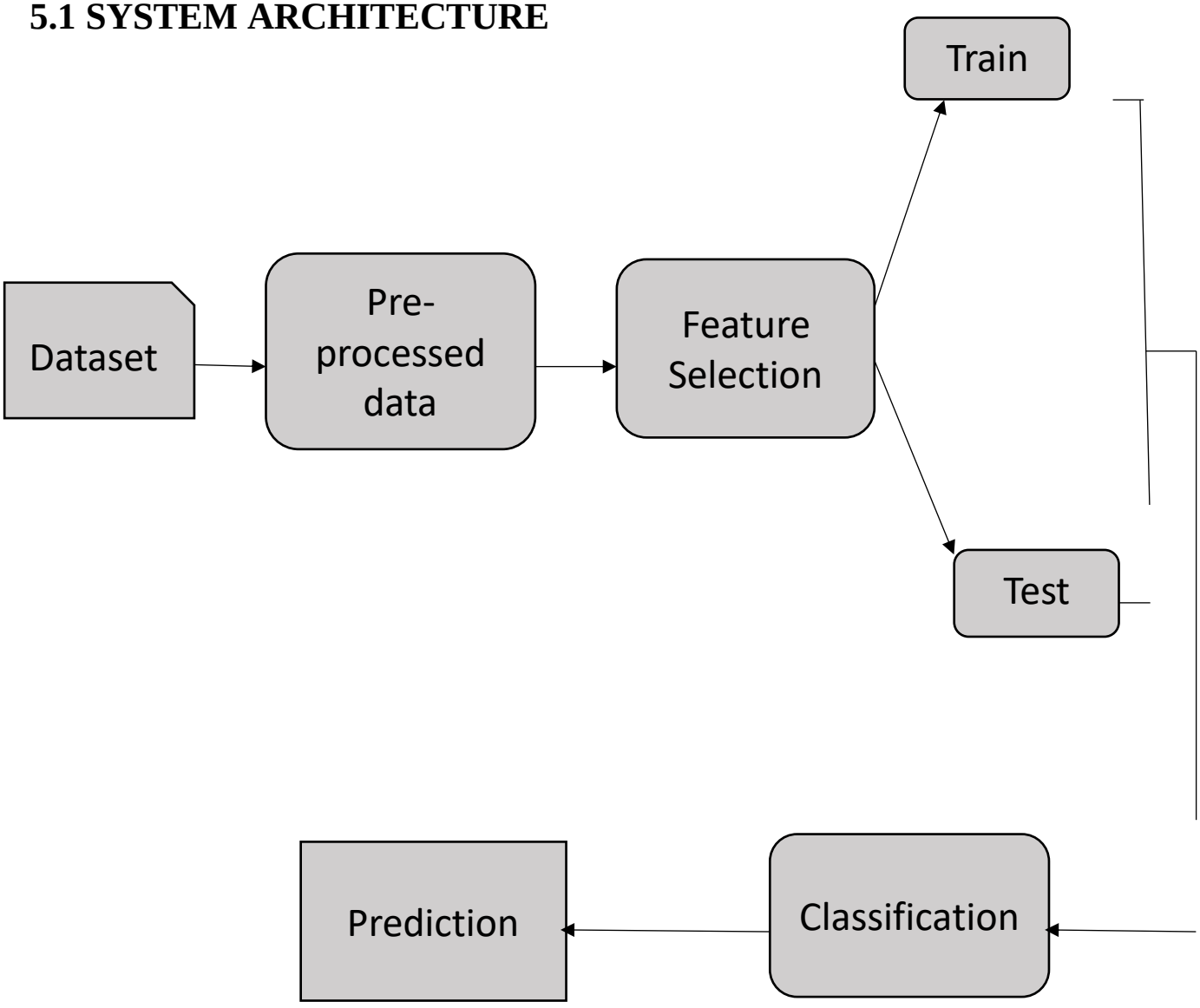


Figure 5.1 : System Architecture

5.2 FLOW DIAGRAM

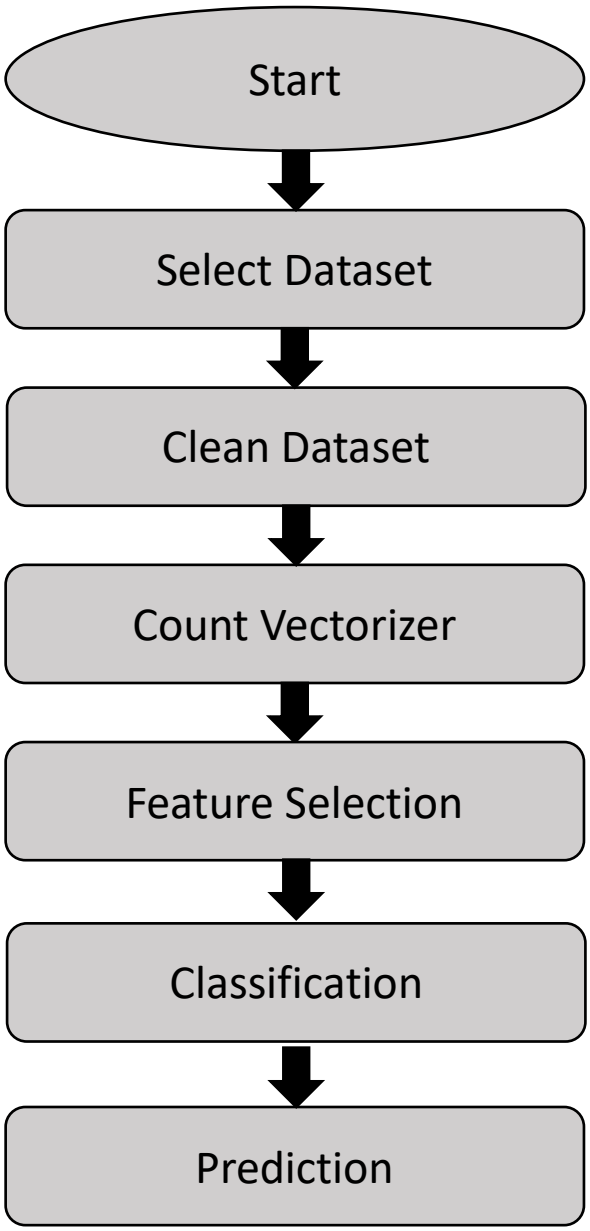


Figure 5.2 : Flow diagram

5.3 USE CASE DIAGRAM

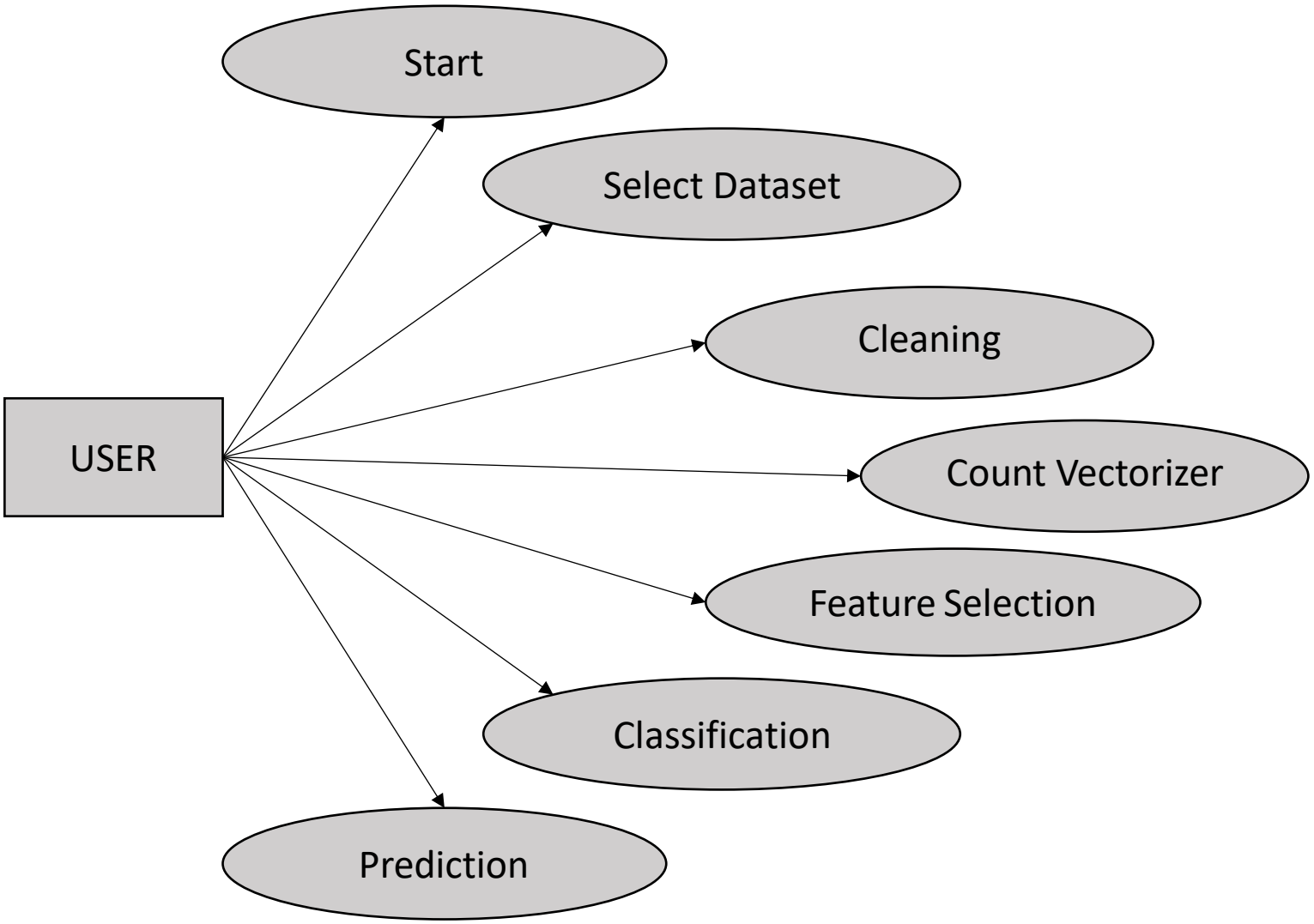


Figure 5.3 : Usecase Diagram

5.4 ER DIAGRAM

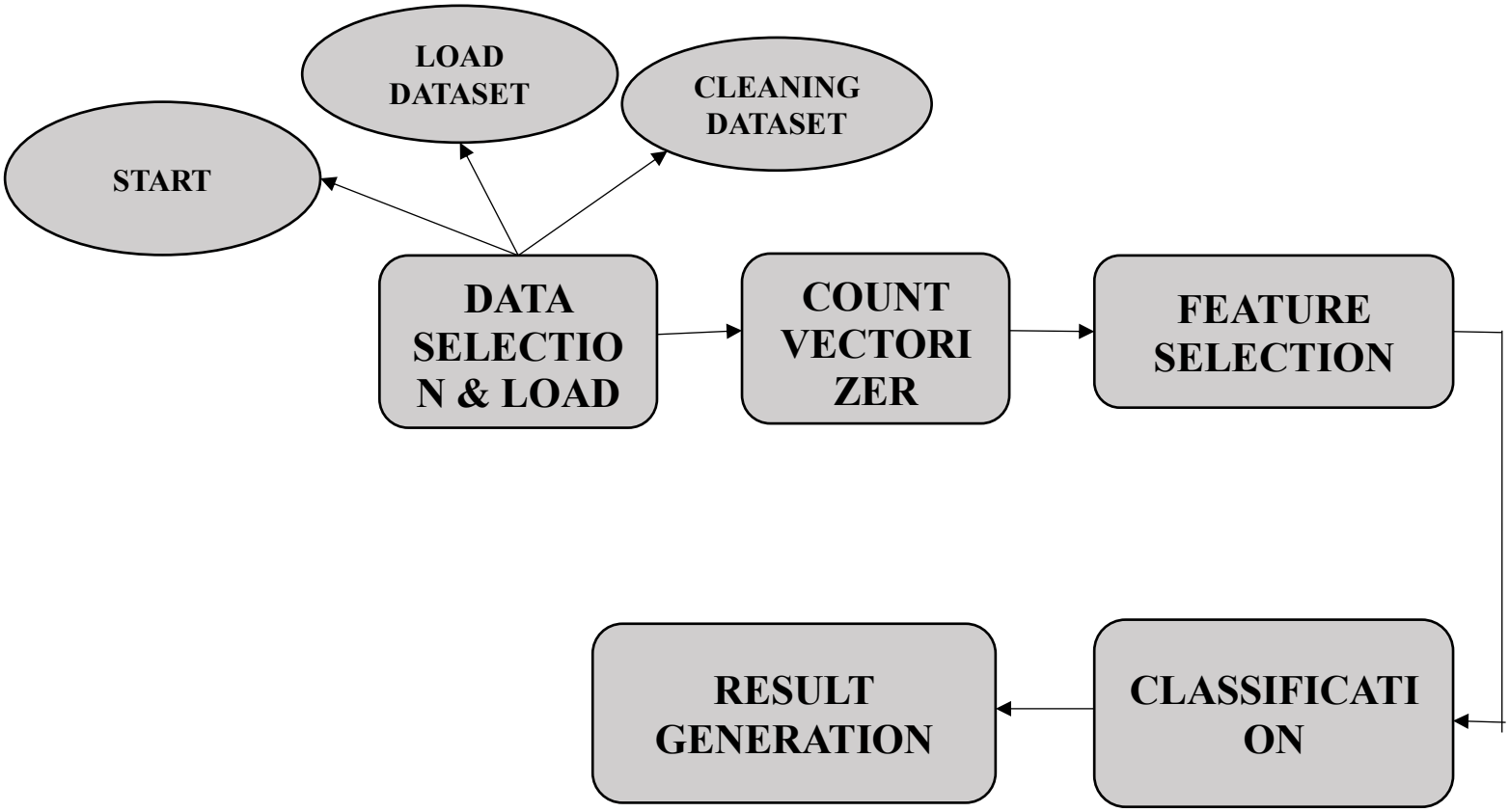


Figure 5.4 : ER Diagram

5.5 SEQUENCE DIAGRAM

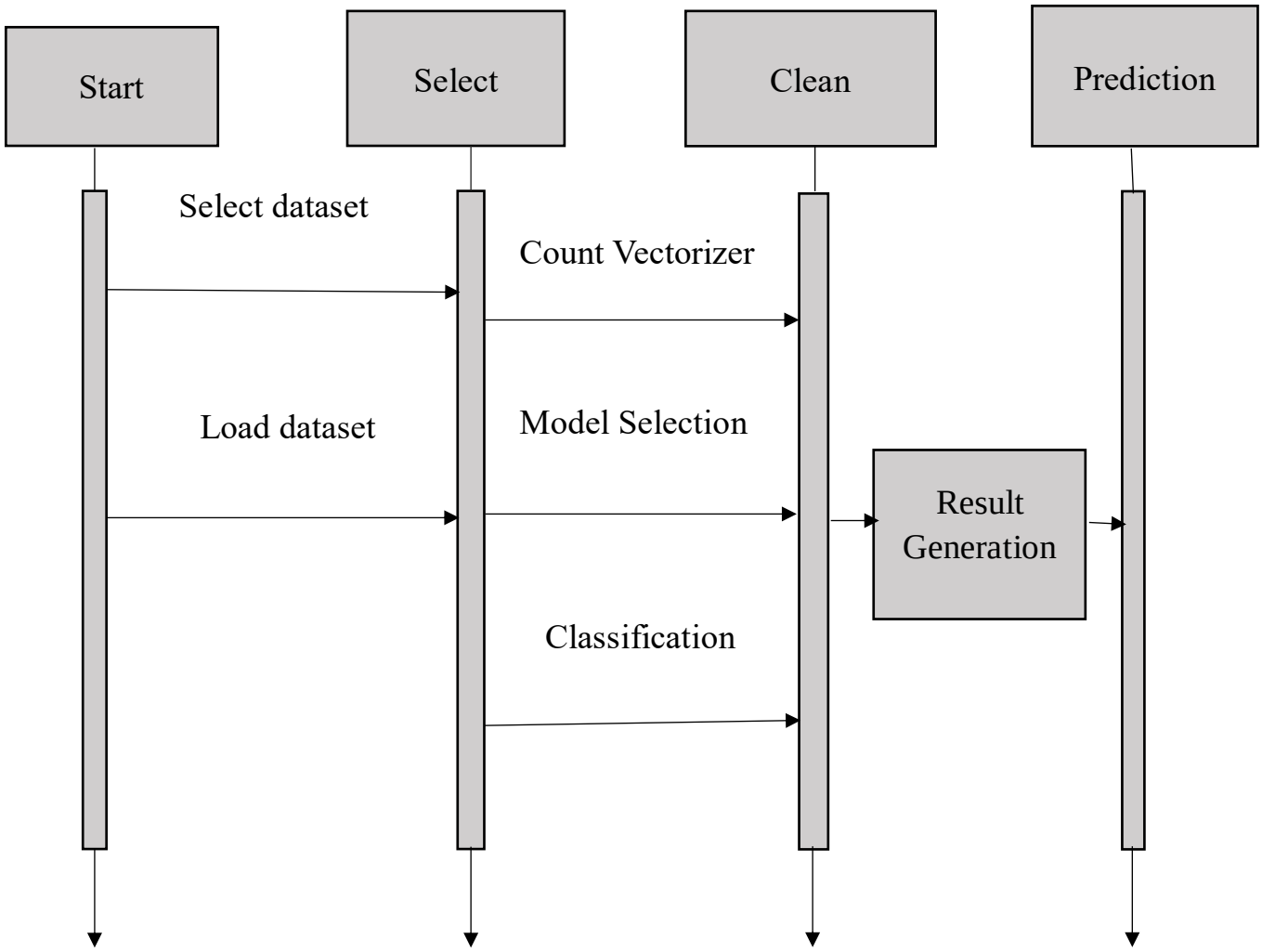


Figure 5.5 : Sequence Diagram

5.6 MODULE DESCRIPTION

5.6.1 MODULES

- Data Selection and Loading
- Data Preprocessing
- Splitting Dataset into Train and Test Data
- Classification
- Prediction
- Result Generation

5.6.2 DATA SELECTION AND LOADING

- Data selection is the process of determining the appropriate data type and source, as well as suitable instruments to collect data.
- Data selection precedes the actual practice of data collection and it is the process where data relevant to the analysis is decided and retrieved from the data collection.
- In this project, the Malware dataset is used for detecting Malware type prediction.

5.6.3 DATA PREPROCESSING

- The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.
- **Missing Data:**
  This situation arises when some data is missing in the data. It can be handled in various ways.

- **Ignore the tuples**:

  This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
- **Fill the Missing values:**

  There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.
- **Encoding Categorical data**: That categorical data is defined as variables with a finite set of label values. That most machine learning algorithms require numerical input and output variables. That an integer and one hot encoding is used to convert categorical data to integer data.
- **Count Vectorizer:** Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token **counts**. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

5.6.4 SPLITTING DATASET INTO TRAIN AND TEST DATA

- Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
- One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
- Separating data into training and testing sets is an important part of evaluating data mining models.
- Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.
- To train any machine learning model irrespective what type of dataset is being used you have to split the dataset into training data and testing data.

5.6.5 CLASSIFICATION

- Classification is the problem of identifying to which of a set of categories, a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

- **Random forests** or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

- **Decision Trees** are a type of Supervised Machine Learning (that is you **explain** what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. An **example** of a **decision tree** can be **explained** using above binary tree.
- The **SVM** is one of the most powerful methods in machine learning algorithms. It can find a balance between model complexity and classification ability given limited sample information. Compared to other machine learning methods, the SVM has many advantages in that it can overcome the effects of noise and work without any prior knowledge. The SVM is a non-probabilistic binary linear classifier that predicts an input to one of two classes for each given input. It optimizes the linear analysis and classification of hyperplane formation techniques.
- **The NN algorithm** is mainly used for classification and regression in machine learning. To determine the category of an unknown sample, all training samples are used as representative points, the distances between the unknown sample and all training sample points are calculated, and the NN is used. The category is the sole basis for determining the unknown sample category.

Because the NN algorithm is particularly sensitive to noise data, the K-nearest neighbour algorithm (KNN) is introduced. The main concept of the KNN is that when the data and tags in the training set are known, the test data are input, the characteristics of the test data are compared with the features corresponding to the training set, and the most similar K in the training set is found.

5.6.6 PREDICTION

Predictive analytics algorithms try to achieve the lowest error possible by either using “boosting” or “bagging”.

Accuracy − Accuracy of classifier refers to the ability of classifier. It predict the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

Speed − Refers to the computational cost in generating and using the classifier or predictor.

Robustness − It refers to the ability of classifier or predictor to make correct predictions from given noisy data.

Scalability − Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.

Interpretability − It refers to what extent the classifier or predictor understands.

5.6.7 RESULT GENERATION

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

- **Accuracy**

  Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers

to how well a given predictor can guess the value of predicted attribute for a new data.

$$AC= \frac{TP+TN}{TP+TN+FP+FN}$$

- **Precision**

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

$$Precision=\frac{TP}{TP+FP}$$

- **Recall**

Recall is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

- **ROC**

ROC curves are frequently used to show in a graphical way the connection/trade-off between clinical sensitivity and specificity for every possible cut-off for a test or a combination of tests. In addition the area under the ROC curve gives an idea about the benefit of using the test(s) in question.

- **Confusion matrix**

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

CHAPTER 6

SOFTWARE DESCRIPTION

6.1 TECHNOLOGIES

6.1.1 Python

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

6.2 FEATURES OF PYTHON

6.2.1 Simple

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

6.2.2 Easy to learn

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

6.3 FREE AND OPEN SOURCE

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

6.4 HIGH LEVEL LANGUAGE

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

6.5 PORTABLE

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, and # -*- coding: utf-8 -*-

z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like Kivy to create games for your computer *and* for iPhone, iPad, and Android.

6.6 INTERPRETED

This requires a bit of explanation.A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

6.7 OBJECT ORIENTED

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

6.8 EXTENSIBLE

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

6.9 EMBEDDABLE

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

6.10 EXTENSIVE LIBRARIES

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

6.11 FEASIBILITY STUDY

The feasibility study is carried out to test whether the proposed system is worth being implemented. The proposed system will be selected if it is best enough in meeting the performance requirements.

The feasibility carried out mainly in three sections namely.

- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility

6.11.1 Economic Feasibility

Economic analysis is the most frequently used method for evaluating effectiveness of the proposed system. More commonly known as cost benefit analysis. This procedure determines the benefits and saving that are expected from the system of the proposed system. The hardware in system department if sufficient for system development.

6.11.2 Technical Feasibility

This study centre around the system's department hardware, software and to what extend it can support the proposed system department is having the required hardware and software there is no question of increasing the cost of implementing the proposed system. The criteria, the proposed system is technically feasible and the proposed system can be developed with the existing facility.

6.11.3 Behavioural Feasibility

People are inherently resistant to change and need sufficient amount of training, which would result in lot of expenditure for the organization. The proposed system can generate reports with day-to-day information immediately at the user's request, instead of getting a report, which doesn't contain much detail.

CHAPTER 7

TESTING AND IMPLEMENTATION

7.1TESTING OF PRODUCT

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to variety of tests-on-line response, Volume Street, recovery and security and usability test. A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

7.2 UNIT TESTING

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during

the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

7.3 INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing.
ii) Bottom-up integration testing.

7.4 WHITE BOX TESTING

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

7.5 BLACK BOX TESTING

- ✓ Black box testing is done to find incorrect or missing function
- ✓ Interface error
- ✓ Errors in external database access
- ✓ Performance errors
- ✓ Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

7.6 VALIDATION TESTING

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

7.7 USER ACCEPTANCE TESTING

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

7.8 OUTPUT TESTING

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be

correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

7.8.1 System Implementation

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

- ✓ The active user must be aware of the benefits of using the system
- ✓ Their confidence in the software built up
- ✓ Proper guidance is impaired to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

7.8.2 User Training

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important. Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

7.8.3 Training on the Application Software

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered.

7.8.4 Operational Documentation

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

7.8.5 System Maintenance

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affect a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only thru proper system maintenance procedures, the system can be adapted to cope up with these changes. Software maintenance is of course, far more than “finding mistakes”.

7.8.6 Corrective Maintenance

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called Corrective Maintenance.

7.8.7 Adaptive Maintenance

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore Adaptive maintenance termed as an activity that modifies software properly with a changing environment is both necessary & common place.

7.8.8 Perceptive Maintenance

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, Perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

7.8.9 Preventive Maintenance

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

7.9 TYPES OF SOFTWARE TESTING

7.9.1 Ad-hoc testing

This type of software testing is very informal and unstructured and can be performed by any stakeholder with no reference to any test case or test design documents. The person performing Ad-hoc testing has a good understanding of the domain and workflows of the application to try to find defects and break the software.

7.9.2 Acceptance Testing

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge.

7.9.3 Accessibility Testing

In accessibility testing, the aim of the testing is to determine if the contents of the website can be easily accessed by disable people. Various checks such as colour and contrast (for colour blind people), font size for visually impaired, clear and concise text that is easy to read and understand.

7.9.4 Agile Testing

Agile Testing is a type of software testing that accommodates agile software development approach and practices. In an Agile development environment, testing is an integral part of software development and is done along with coding. Agile testing allows incremental and iterative coding and testing.

7.9.5 API Testing

API testing is a type of testing that is similar to unit testing. Each of the Software APIs are tested as per API specification. API testing is mostly done by testing team unless APIs to be tested or complex and needs extensive coding. API testing requires understanding both API functionality and possessing good coding skills.

7.9.6 Automated testing

This is a testing approach that makes use of testing tools and/or programming to run the test cases using software or custom developed test utilities. Most of the automated tools provided capture and playback facility, however there are tools that require writing extensive scripting or programming to automate test cases.

7.9.7 All Pairs testing

Also known as Pair wise testing, is a black box testing approach and a testing method where in for each input is tested in pairs of inputs, which helps to test software works as expected with all possible input combinations.

7.9.8 Beta Testing

This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.

7.9.9 Black Box testing

Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing

method relies on testing software with various inputs and validating results against expected output.

7.910 Backward Compatibility Testing

Type of software testing performed to check newer version of the software can work successfully installed over previous version of the software and newer version of the software works as fine with table structure, data structures, files that were created by previous version of the software.

7.9.11 Boundary Value Testing (BVT)

Boundary Value Testing is a testing technique that is based on concept "error aggregates at boundaries". In this testing technique, testing is done extensively to check for defects at boundary conditions. If a field accepts value 1 to 100 then testing is done for values 0, 1, 2, 99, 100 and 101.

7.9.12 Big Bang Integration testing

This is one of the integration testing approaches, in Big Bang integration testing all or all most all of the modules are developed and then coupled together.

7.9.13 Bottom up Integration testing

Bottom up integration testing is an integration testing approach where in testing starts with smaller pieces or sub systems of the software till all the way up covering entire software system.

7.9.14 Branch Testing

Is a white box testing method for designing test cases to test code for every branching condition? Branch testing method is applied during unit testing.

7.9.15 Browser compatibility Testing

It is one of the sub types of testing of compatibility testing performed by testing team. Browser compatibility testing is performed for web applications with combination of different browsers and operating systems.

7.9.16 Compatibility testing

Compatibility testing is one of the test types performed by testing team. Compatibility testing checks if the software can be run on different hardware, operating system, bandwidth, databases, web servers, application servers, hardware peripherals, emulators, different configuration, processor, different browsers and different versions of the browsers etc.

7.9.17 Component Testing

This type of software testing is performed by developers. Component testing is carried out after completing unit testing. Component testing involves testing a group of units as code together as a whole rather than testing individual functions, methods.

7.9.18 Condition Coverage Testing

Condition coverage testing is a testing technique used during unit testing, where in developer tests for all the condition statements like if, if else, case etc., in the code being unit tested.

7.9.19 Dynamic Testing

Testing can be performed as Static Testing and Dynamic testing, Dynamic testing is a testing approach where-in testing can be done only by executing code or software are classified as Dynamic Testing. Unit testing, Functional testing, regression testing, performance testing etc.

7.10 DECISION COVERAGE TESTING

Is a testing technique that is used in Unit testing, objective of decision coverage testing is to expertise and validate each and every decisions made in the code e.g. if, if else, case statements.

7.10.1 End-to-end Testing

End to end testing is performed by testing team, focus of end to end testing is to test end to end flows e.g. right from order creation till reporting or order creation till item return etc. and checking. End to end testing is usually focused mimicking real life scenarios and usage. End to end testing involves testing information flow across applications.

7.10.2 Exploratory Testing

Exploratory testing is an informal type of testing conducted to learn the software at the same time looking for errors or application behaviour that seems non-obvious. Exploratory testing is usually done by testers but can be done by other stake holders as well like Business Analysts, developers, end users etc. who are interested in learning functions of the software and at the same time looking for errors or behaviour is seems non-obvious.

7.10.3 Equivalence Partitioning

Equivalence partitioning is also known as Equivalence Class Partitioning is a software testing technique and not a type of testing by itself. Equivalence partitioning technique is used in black box and grey box testing types. Equivalence partitioning classifies test data into Equivalence classes as positive Equivalence classes and negative Equivalence classes, such classification ensures both positive and negative conditions are tested.

7.10.2 Functional Testing

Functional testing is a formal type of testing performed by testers. Functional testing focuses on testing software against design document, Use cases and requirements document. Functional testing is a black box type of testing and does not require internal working of the software unlike white box testing.

7.10.3 Fuzz Testing

Fuzz testing or fuzzing is a software testing technique that involves testing with unexpected or random inputs. Software is monitored for failures or error messages that are presented due to the input errors.

7.10.4 GUI (Graphical User Interface) testing

This type of software testing is aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mock-ups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mock-ups, detailed design documents and functional requirements

7.10.5 Glass box Testing

Glass box testing is another name for White box testing. Glass box testing is a testing method that involves testing individual statements, functions etc., Unit testing is one of the Glass box testing methods.

7.10.6 Gorilla Testing

This type of software testing is done by software testing team, has a scary name though? Objective of Gorilla Testing is to exercise one or few functionality thoroughly or exhaustively by having multiple people test the same functionality.

7.10.7 Happy Path Testing

Also known as Golden path testing, this type of testing focuses on selective execution of tests that do not exercise the software for negative or error conditions.

7.10.8 Integration Testing

Integration testing also known as met in short, in one of the important types of software testing. Once the individual units or components are tested by developers as working then testing team will run tests that will test the connectivity among these units/component or multiple units/components. There are different approaches for Integration testing namely, Top-down integration testing, Bottom-up integration testing and a combination of these two known as Sand witch testing.

7.10.9 Interface Testing

Software provides support for one or more interfaces like “Graphical user interface”, “Command Line Interface” or “Application programming interface” to interact with its users or other software. Interfaces serves as medium for software to accept input from user and provide result. Approach for interface testing depends on the type of the interface being testing like GUI or API or CLI.

7.10.10 Internationalization Testing

Internationalization testing is a type of testing that is performed by software testing team to check the extent to which software can support Internationalization i.e., usage of different languages, different character sets, double byte characters etc., For e.g.: Gmail, is a web application that is used by people all over work with different languages, single by or multi byte character sets.

7.10.11 Keyword-driven Testing

Keyword driver testing is more of an automated software testing approach than a type of testing itself. Keyword driven testing is known as action driven testing or table driven testing.

7.10.12 Load Testing

Load testing is a type of non-functional testing; load testing is done to check the behaviour of the software under normal and over peak load conditions. Load testing is usually performed using automated testing tools. Load testing intends to find bottlenecks or issues that prevent software from performing as intended at its peak workloads.

7.10.13 Localization Testing

Localization testing a type of software testing performed by software testers, in this type of testing, software is expected to adapt to a particular locale, it should support a particular locale/language in terms of display, accepting input in that particular locale, display, font, date time, currency etc., related to a particular locale. For e.g. many web applications allow choice of locale like English, French, German or Japanese. So once locale is defined or set in the

7.10.14 Negative Testing

This type of software testing approach, which calls out the “attitude to break”, these are functional and non-functional tests that are intended to break the software by entering incorrect data like incorrect date, time or string or upload binary file when text files supposed to be upload or enter huge text string for input fields etc. It is also a positive test for an error condition.

7.10.15 Non-functional testing

Software are built to fulfil functional and non-functional requirements, non-functional requirements like performance, usability, localization etc., There are many types of testing like compatibility testing, compliance testing, localization testing, usability testing, volume testing etc., that are carried out for checking non-functional requirements.

7.10.16 Pair Testing

It is a software testing technique that can be done by software testers, developers or Business analysts (BA). As the name suggests, two people are paired together, one to test and other to monitor and record test results. Pair testing can also be performed in combination of tester-developer, tester-business analyst or developer-business analyst combination. Combining testers and developers in pair testing helps to detect defects faster, identify root cause, fix and test the fix.

7.10.17 Performance Testing

It is a type of software testing and part of performance engineering that is performed to check some of the quality attributes of software like Stability, reliability, availability. Performance testing is carried out by performance engineering team. Unlike Functional testing, Performance testing is done to check non-functional requirements. Performance testing checks how well software

works in anticipated and peak workloads. There are different variations or sub types of performance like load testing, stress testing, volume testing, soak testing and configuration testing.

7.10.18 Penetration Testing

It is a type of security testing, also known as pen test in short. Penetration testing is done to tests how secure software and its environments (Hardware, Operating system and network) are when subject to attack by an external or internal intruder. Intruder can be a human/hacker or malicious programs. Penetration Testing is a way of ethical hacking, an experienced Penetration tester will use the same methods and tools that a hacker would use but the intention of Penetration tester is to identify vulnerability and get them fixed before a real hacker or malicious program exploits it.

7.10.19 Regression Testing

It is a type of software testing that is carried out by software testers as functional regression tests and developers as Unit regression tests. Objective of regression tests are to find defects that got introduced to defect fix (is) or introduction of new feature(s). Regression tests are ideal candidate for automation.

7.10.20 Retesting

It is a type of retesting that is carried out by software testers as a part of defect fix verification. For e.g. a tester is verifying a defect fix and let us say that there are 3 test cases failed due to this defect. Once tester verifies defect fix as resolved, test will retest or test the same functionality again by executing the test cases that were failed earlier.

7.11 RISK BASED TESTING

It is a type of software testing and a different approach towards testing a software. In Risk based testing, requirements and functionality of software to be tested are prioritized as Critical, High, Medium and low. In this approach, all critical and high priority tests are tested and them followed by Medium. Low priority or low risk functionality are tested at the end or may not base on the time available for testing.

7.11.1 Smoke testing

It is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e., major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find “show stopper” defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as build verification test.

7.11.2 Security Testing

It is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.

7.11.3 Sanity Testing

It is a type of testing that is carried out mostly by testers and in some projects by developers as well. Sanity testing is a quick evaluation of the software, environment, network, external systems are up & running, software environment as a whole is stable enough to proceed with extensive testing. Sanity tests are narrow and most of the time sanity tests are not documented.

7.11.4 Scalability Testing

It is a non-functional test intended to test one of the software quality attributes i.e. “Scalability”. Scalability test is not focused on just one or few functionality of the software instead performance of software as a whole. Scalability testing is usually done by performance engineering team. Objective of scalability testing is to test the ability of the software to scale up with increased users, increased transactions, increase in database size etc., It is not necessary that software’s performance increases with increase in hardware configuration, scalability tests helps to find out how much more workload the software can support with expanding user base, transactions, data storage etc.,

7.11.5 Stability Testing

It is a non-functional test intended to test one of the software quality attributes i.e. “Stability”. Stability testing focuses on testing how stable software is when it is subject to loads at acceptable levels, peak loads, loads generated in spikes, with more volumes of data to be processed. Scalability testing will involve performing different types of performance tests like load testing, stress testing, spike testing, soak testing, spike testing etc…

7.11.6 Static Testing

Static testing is a form of testing where in approaches like reviews, walkthroughs are employed to evaluate the correctness of the deliverable. In static testing software code is not executed instead it is reviewed for syntax, commenting, naming convention, size of the functions and methods etc. Static testing usually has check lists against which deliverables are evaluated. Static testing can be applied for requirements, designs, and test cases by using approaches like reviews or walkthroughs.

7.11.7 Stress Testing

Stress testing a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behaviour of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

We reviewed several influential algorithms for malware prediction based on various machine learning techniques. Characteristics of ML techniques makes it possible to design IDS that have high prediction rates and low false positive rates while the system quickly adapts itself. We divided these algorithms into three types of ML-based classifiers: Random Forest (RF), Support vector machine(SVM), and Decision Tree (DT). Although these two algorithms share many similarities, several features of techniques, such as adaptation, high computational speed and error resilience in the face of noisy information, conform the requirement of building efficient software quality prediction.

8.2 FUTURE WORK

In future, it is possible to provide extensions or modifications to the proposed clustering and classification algorithms using intelligent agents to achieve further increased performance. Apart from the experimented combination of data mining techniques, further combinations such as artificial intelligence, soft computing and other clustering algorithms can be used to improve the detection accuracy and to reduce the rate of false negative alarm and false positive alarm. Finally, the software quality prediction system can be extended as a software fault prevention system to enhance the performance of the system.

APPENDIX I

```
import tkinter
import tkinter.messagebox
import sqlite3
from tkinter import *
import tkinter as tk
from random import *
import string
entry_1 = None;
entry_2 = None;
entry_3 = None;
class For Frames(t k .Tk):
def __init__(self, *args, **kwargs):
tk.Tk.__init__(self, *args, **kwargs)
container = tk.Frame(self)
container.pack(side="top", fill="both", expand=True)
container.grid_rowconfigure(0, weight=1)
container.grid_columnconfigure(0, weight=1)
self.frames = {}
for F in (Registerform,Login):
page_name = F.__name__
frame = F(parent=container, controller=self)
self.frames[page_name] = frame
frame.grid(row=0, column=0, sticky="nsew")
self.show_frame("Registerform")
def show_frame(self, page_name):
frame = self.frames[page_name]
    frame.tkraise()
class Registerform(tk.Frame):
```

```
import warnings
warnings.filterwarnings("ignore")
print("==================================================")
print("DDOS Dataset")
print(" Process - DDOS Attack Detection")
print("==================================================")
##1.data slection----------------------------------------------
#def main():
dataframe=pd.read_csv("dataset.csv")
print("------------------------------------------")
print()
print("Data Selection")
print("Samples of our input data")
print(dataframe.head(10))
print("------------------------------------------")
print()
#2.pre processing---------------------------------------------
#checking  missing values
print("------------------------------------------")
print()
print("Before Handling Missing Values")
print()
print(dataframe.isnull().sum())
print("------------------------------------------")
print()
  print("------------------------------------------")
print("After handling missing values")
print()
dataframe_2=dataframe.fillna(0)
```

```
print(dataframe_2.isnull().sum())
print()
print("------------------------------------------------")
#label encoding
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
print("------------------------------------------------")
print("Before Label Handling ")
print()
print(dataframe_2.head(10))
print("------------------------------------------------")
print()
#3.Data splitting-----------------------------------------------
df_train_y=dataframe_2["label"]
df_train_X=dataframe_2.iloc[:,:20]
from sklearn.preprocessing import LabelEncoder
number = LabelEncoder()
df_train_X['proto'] = number.fit_transform(df_train_X['proto'].astype(str))
df_train_X['service'] = number.fit_transform(df_train_X['service'].astype(str))
df_train_X['state'] = number.fit_transform(df_train_X['state'].astype(str))
#df_train_X['attack_cat'] =
number.fit_transform(df_train_X['attack_cat'].astype(str))
print("================================================")
print(" Preprocessing")
print("================================================")
df_train_X.head(5)
x=df_train_X
y=df_train_y
##4.feature selection-----------------------------------------
```

```
##kmeans
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
x, y_true = make_blobs(n_samples=175341, centers=2,cluster_std=0.30,
random_state=0)
plt.scatter(x[:, 0], x[:, 1], s=20);
kmeans = KMeans(n_clusters=2)
kmeans.fit(x)
y_kmeans = kmeans.predict(x)
plt.scatter(x[:, 0], x[:, 1], c=y_kmeans, s=20, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
plt.title("k-means")
plt.show()
#-------------------------------------------------------------------------------
x_train,x_test,y_train,y_test = train_test_split(df_train_X,y,test_size =
0.20,random_state = 42)
print(x_train.shape )
print(x_test.shape )
print(y_train.shape )
print(y_test.shape )
from sklearn.tree import DecisionTreeClassifier
model_DD = DecisionTreeClassifier()
max_depth = range(1,10,1)
min_samples_leaf = range(1,10,2)
tuned_parameters = dict(max_depth=max_depth,
min_samples_leaf=min_samples_leaf)
from sklearn.model_selection import GridSearchCV
```

```
DD = GridSearchCV(model_DD, tuned_parameters,cv=10)
DD.fit(x_train, y_train)
print("Best: %f using %s" % (DD.best_score_, DD.best_params_))
import numpy as np
y_prob = DD.predict_proba(x_test)[:,1]
y_pred = np.where(y_prob > 0.5, 1, 0)
DD.score(x_test, y_pred)
print()
print("----------------------------------------------------------------------")
print("enclarge c4  Algorithm ")
print()
Result_2=accuracy_score(y_test, y_pred)*100
print(metrics.classification_report(y_test,y_pred))
print()
print("enclarge c4  Algorithm Accuracy is:",Result_2,'%')
print()
print("Confusion Matrix:")
from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(y_test, y_pred)
print(cm1)
print("------------------------------------------------------")
print()
#-------------------------------------------------------------------------------------------
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = "gini", random_state =
100,max_depth=3, min_samples_leaf=5)
dt.fit(x_train, y_train)
dt_prediction=dt.predict(x_test)
print()
```

```
print("-----------------------------------------------------------------")
print("Decision Tree")
print()
Result_2=accuracy_score(y_test, dt_prediction)*100
print(metrics.classification_report(y_test,dt_prediction))
print()
print("DT Accuracy is:",Result_2,'%')
print()
print("Confusion Matrix:")
from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(y_test, dt_prediction)
print(cm1)
print("-----------------------------------------------------")
print()
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(cm1, annot = True, cmap ='plasma',
        linecolor ='black', linewidths = 1)
plt.show()
#ROC graph
import pickle
with open('model1.pkl','wb') as files:
    pickle.dump(dt,files)
#-------------------------------------------------------------------------
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
from sklearn.ensemble import GradientBoostingClassifier
gradient_booster = GradientBoostingClassifier(learning_rate=0.1)
gradient_booster.get_params()
```

```
gradient_booster.fit(x_train,y_train)
gb_prediction = gradient_booster.predict(x_test)
print(classification_report(y_test,gradient_booster.predict(x_test)))
Result_2=accuracy_score(y_test, gb_prediction)*100
print()
print("gradient_booster Accuracy is:",Result_2,'%')
print()
print("Confusion Matrix:")
from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(y_test, dt_prediction)
print(cm1)
print("----------------------------------------------------")
print()
#------------------------------------------------------------------------
"Navie Bayies "
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(x_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print("Navie Bayies Accuracy is:",Result_2,'%')
print()
print("Confusion Matrix:")
from sklearn.metrics import confusion_matrix
cm1=confusion_matrix(y_test, y_pred)
```

```
print(cm1)
print("-----------------------------------------------------")
print()
#-------------------------------------------------------------------------------------------

from easygui import *
Key = "Enter the DDOS   Id to be Search"
# window title
title = "DDOS Id "
# creating a integer box
str_to_search1 = enterbox(Key, title)
input = int(str_to_search1)
import tkinter as tk
if (y_pred[input] ==0 ):
    print("Non Attack ")
    root = tk.Tk()
    T = tk.Text(root, height=20, width=30)
    T.pack()
    T.insert(tk.END, "Non Attack ")
    tk.mainloop()
elif (y_pred[input] ==1 ):
    print("Attack ")
    root = tk.Tk(    T = tk.Text(root, height=20, width=30) T.pack(
T.insert(tk.END, "Attack"tk.mainloop()
```

APPENDIX II

```
================================================
Malware Dataset
 Process - Malware Attack Detection
================================================
------------------------------------------------

Data Selection
Samples of our input data
   id       dur proto service  ... ct_srv_dst  is_sm_ips_ports  attack_cat  label
0   1  0.121478   tcp       -  ...          1                0      Normal      0
1   2  0.649902   tcp       -  ...          6                0      Normal      0
2   3  1.623129   tcp       -  ...          6                0      Normal      0
3   4  1.681642   tcp     ftp  ...          1                0      Normal      0
4   5  0.449454   tcp       -  ...         39                0      Normal      0
5   6  0.380537   tcp       -  ...         39                0      Normal      0
6   7  0.637109   tcp       -  ...         39                0      Normal      0
7   8  0.521584   tcp       -  ...         39                0      Normal      0
8   9  0.542905   tcp       -  ...         39                0      Normal      0
9  10  0.258687   tcp       -  ...         39                0      Normal      0
```

Figure 8.1: Data Selection

```
------------------------------------------------
After handling missing values

id                  0
dur                 0
proto               0
service             0
state               0
spkts               0
dpkts               0
sbytes              0
dbytes              0
rate                0
sttl                0
dttl                0
sload               0
dload               0
sloss               0
dloss               0
sinpkt              0
```

figure 8.2 : Data Preprocessing

```
------------------------------------------------
Before Label Handling

   id       dur proto service  ... ct_srv_dst  is_sm_ips_ports  attack_cat  label
0   1  0.121478   tcp       -  ...          1                0      Normal      0
1   2  0.649902   tcp       -  ...          6                0      Normal      0
2   3  1.623129   tcp       -  ...          6                0      Normal      0
3   4  1.681642   tcp     ftp  ...          1                0      Normal      0
4   5  0.449454   tcp       -  ...         39                0      Normal      0
5   6  0.380537   tcp       -  ...         39                0      Normal      0
6   7  0.637109   tcp       -  ...         39                0      Normal      0
7   8  0.521584   tcp       -  ...         39                0      Normal      0
8   9  0.542905   tcp       -  ...         39                0      Normal      0
9  10  0.258687   tcp       -  ...         39                0      Normal      0

[10 rows x 45 columns]
------------------------------------------------
```

Figure 8.3 : Data label Encoding

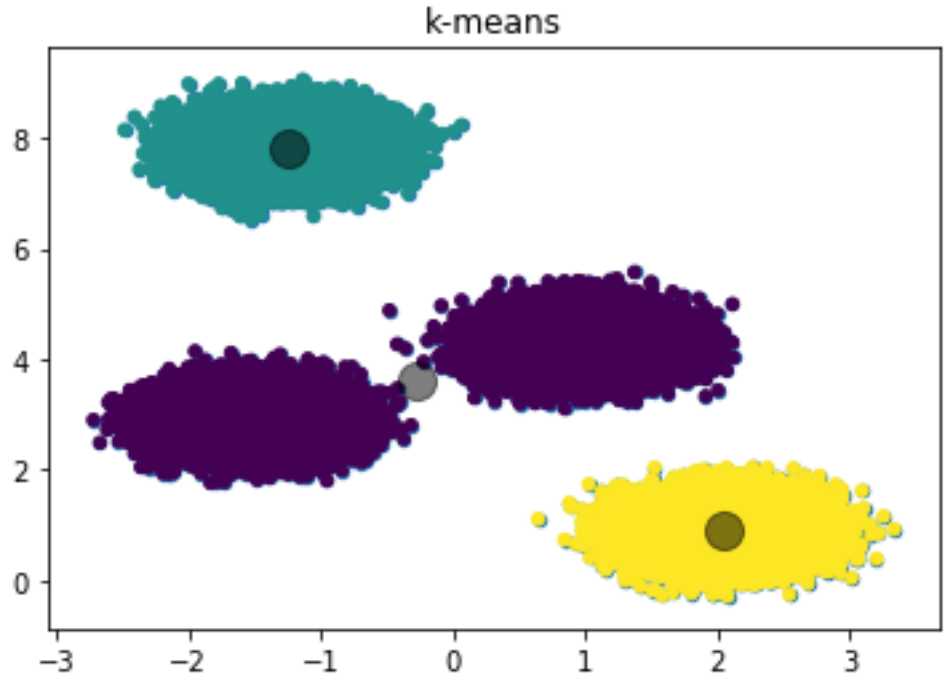
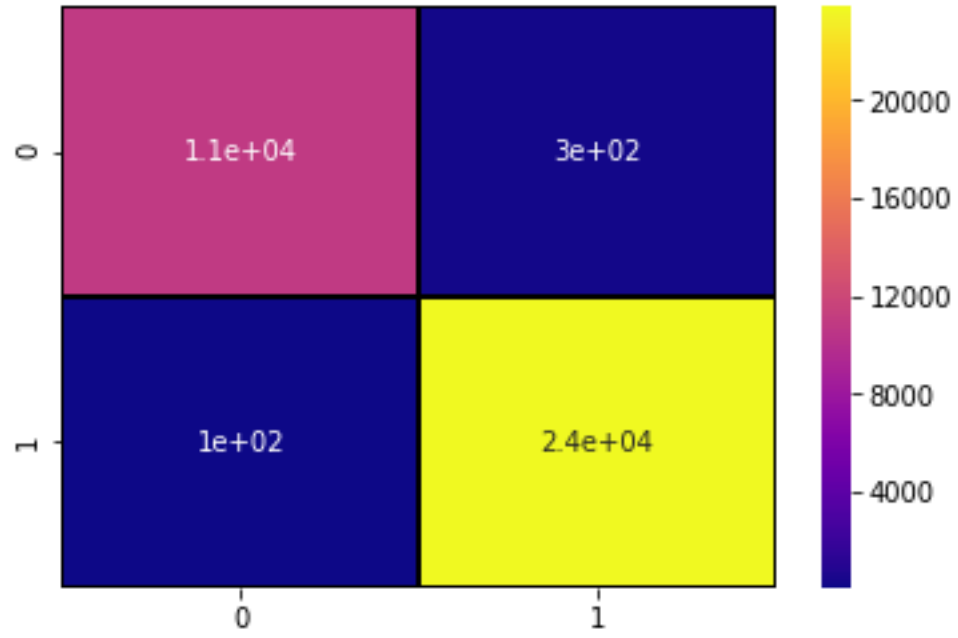


Figure 8.4 : K mean clustering

```
------------------------------------------------------------
Random Forest

              precision    recall  f1-score   support

           0       0.99      0.97      0.98     11169
           1       0.99      1.00      0.99     23900

   micro avg       0.99      0.99      0.99     35069
   macro avg       0.99      0.98      0.99     35069
weighted avg       0.99      0.99      0.99     35069


Random Forest Accuracy is: 98.85083692149762 %

Confusion Matrix:
[[10868   301]
 [  102 23798]]
------------------------------------------------------------
```

Figure 8.5 : Random forest




```
------------------------------------------------------------

Navie Bayies Accuracy is: 97.51632495936582 %

Confusion Matrix:
[[ 9417  1752]
 [  891 23009]]
------------------------------------------------------------
```

Figure 8.6 Navie bayies

```
--------------------------------------------------------------------
Decision Tree

              precision    recall  f1-score   support

           0       1.00      0.89      0.94     11169
           1       0.95      1.00      0.98     23900

   micro avg       0.97      0.97      0.97     35069
   macro avg       0.98      0.95      0.96     35069
weighted avg       0.97      0.97      0.97     35069


DT Accuracy is: 96.60098662636517 %

Confusion Matrix:
[[ 9977  1192]
 [    0 23900]]
------------------------------------------------------------
```

Figure 8.7 Decision Tree

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96     11169
           1       0.97      1.00      0.98     23900

    micro avg       0.98      0.98      0.98     35069
    macro avg       0.98      0.96      0.97     35069
weighted avg       0.98      0.98      0.97     35069


gradient_booster Accuracy is: 97.51632495936582 %

Confusion Matrix:
[[ 9977  1192]
 [    0 23900]]
```

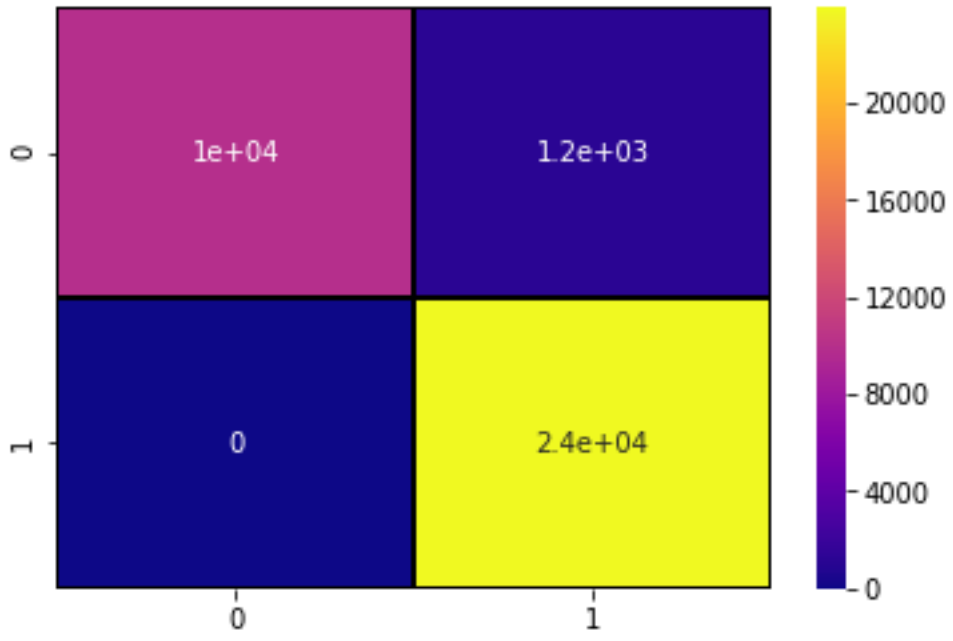
Figure 8.8 : Gradient Booster

```
assification Report:
              precision    recall  f1-score   support

           0       0.96      0.88      0.92     11169
           1       0.95      0.98      0.96     23900

 micro avg       0.95      0.95      0.95     35069
 macro avg       0.95      0.93      0.94     35069
ighted avg       0.95      0.95      0.95     35069

curacy: 0.9510108642961019
```

Figure 8.9 : SVM

```
Enter the Malware Type2
Ransomware
```

Figure 8.10 : Malware Prediction



REFERENCES

- A. A. Khan, M. H. Rehmani, and M. Reisslein, ‘‘Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols,’’ IEEE Commun. Surveys Tuts., vol. 18, no. 1, pp. 860–898, 1st Quart., 2016.
- Y. Lin, X. Zhu, Z. Zheng, Z. Dou, and R. Zhou, ‘‘The individual identification method of wireless device based on dimensionality reduction,’’ J. Supercomput., vol. 75, no. 6, pp. 3010–3027, Jun. 2019.
- T. Liu, Y. Guan, and Y. Lin, ‘‘Research on modulation recognition with ensemble learning,’’ EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 179, 2017.
- Y. Tu, Y. Lin, J. Wang, and J.-U. Kim, ‘‘Semi-supervised learning with generative adversarial networks on digital signal modulation classification,’’ Comput. Mater. Continua, vol. 55, no. 2, pp. 243–254, 2018.
- C. Shi, Z. Dou, Y. Lin, and W. Li, ‘‘Dynamic threshold-setting for RFpowered cognitive radio networks in non-Gaussian noise,’’ EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 192, Nov. 2017.
- Z. Zhang, X. Guo, and Y. Lin, ‘‘Trust management method of D2D communication based on RF fingerprint identification,’’ IEEE Access, vol. 6, pp. 66082–66087, 2018.
- H. Wang, J. Li, L. Guo, Z. Dou, Y. Lin, and R. Zhou, ‘‘cFractal complexitybased feature extraction algorithm of communication signals,’’ Fractals, vol. 25, no. 4, pp. 1740008-1–1740008-3, Jun. 2017.
- J. Zhang, S. Chen, X. Mu, and L. Hanzo, ‘‘Evolutionary-algorithm-assisted joint channel estimation and turbo multiuser detection/decoding for

OFDM/SDMA,'' IEEE Trans. Veh. Technol., vol. 63, no. 3, pp. 1204–1222, Mar. 2014.

- J. Zhang, S. Chen, X. Guo, J. Shi, and L. Hanzo, ''Boosting fronthaul capacity: Global optimization of power sharing for centralized radio access network,'' IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 1916–1929, Feb. 2019.
- J. Wen, Z. Zhou, Z. Liu, M.-J. Lai, and X. Tang, ''Sharp sufficient conditions for stable recovery of block sparse signals by block orthogonal matching pursuit,'' Appl. Comput. Harmon. Anal., vol. 47, pp. 948–974, Nov. 2019.
- J. Wen and X.-W. Chang, ''On the KZ reduction,'' IEEE Trans. Inf. Theory, vol. 65, no. 3, pp. 1921–1935, Mar. 2019.
- Z. Dou, C. Shi, Y. Lin, and W. Li, ''Modeling of non-gaussian colored noise and application in cr multi-sensor networks,'' EURASIP J. Wireless Commun. Netw., vol. 2017, no. 1, p. 192, Nov. 2017.
- M. Liu, J. Zhang, Y. Lin, Z. Wu, B. Shang, and F. Gong, ''Carrier frequency estimation of time-frequency overlapped MASK signals for underlay cognitive radio network,'' IEEE Access, vol. 7, pp. 58277–58285, 2019.
- Y. Lin, Y. Li, X. Yin, and Z. Dou, ''Multisensor fault diagnosis modeling based on the evidence theory,'' IEEE Trans. Rel., vol. 67, no. 2, pp. 513–521, Jun. 2018.
- H. Wang, L. Guo, Z. Dou, and Y. Lin, ''A new method of cognitive signal recognition based on hybrid information entropy and DS evidence theory,'' Mobile Netw. Appl., vol. 23, no. 4, pp. 677–685, Aug. 2018.
- W. Wei and J. M. Mendel, ''Maximum-likelihood classification for digital amplitude-phase modulations,'' IEEE Trans. Commun., vol. 48, no. 2, pp. 189–193, Feb. 2000.

- A. Swami and B. M. Sadler, ‘‘Hierarchical digital modulation classification using cumulants,’’ IEEE Trans. Commun., vol. 48, no. 3, pp. 416–429, Mar. 2000.
- N. Mahmoudi and E. Duman, ‘‘Detecting credit card fraud by modified Fisher discriminant analysis,’’ Expert Syst. Appl., vol. 42, no. 5, pp. 2510–2516, Apr. 2015.
- L. Wu, C. Shen, and A. van den Hengel, ‘‘Deep linear discriminant analysis on Fisher networks: A hybrid architecture for person re-identification,’’ Pattern Recognit., vol. 65, pp. 238–250, May 2017.
- S. Srivastava and M. R. Gupta, ‘‘Distribution-based Bayesian minimum expected risk for discriminant analysis,’’ in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2006, vol. 1, no. 1, pp. 2294–2298.