Improving Ransomware Detection Techniques using Machine Learning

A Project Report

Submitted by

Ishan Chaudhari (RA1711020010096)

VARSHINI V (RA1711020010140)

Under the Guidance of

Dr. Usha G

(Assistant Professor, Department of Software Engineering)

In partial fulfillment of the Requirements for the Degree

of

BACHELOR OF TECHNOLOGY

in

SOFTWARE ENGINEERING

DEPARTMENT OF SOFTWARE ENGINEERING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

Kattankulathur, Kancheepuram District - 603203

May 2021

BONAFIDE CERTIFICATE

This is to certify that this project report titled "**Improving Ransomware Detection Techniques using Machine Learning**" is the bonafide work of **Ishan Chaudhari (RA1711020010096), Varshini V. (RA1711020010140)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. Usha G.
PROJECT GUIDE
Assistant Professor
Dept. of software engineering

Dr. C. Lakshmi
HEAD OF THE DEPARTMENT
Dept. of software engineering

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

A challenge that governments, enterprises as well as individuals are constantly facing is the growing threat of ransomware attacks. Ransomware is a type of malware that encrypts the user's files and then demands a huge sum of money from the user. This increasing complexity calls for more advancement and innovative ideas in defensive strategies used to tackle the problems. In this paper, firstly we discuss the existing research in the field of ransomware detection techniques and their shortcomings. Secondly, a juxtaposed study on various machine learning algorithms to detect ransomware attacks is explained. Various behavioural data such as API Calls, Target files, Registry Operations, Signature, Network Accesses were collected for each ransomware and benign sample. These collected samples were used to train Machine Learning Algorithms like KNN, Naïve Bayes, Random Forest, Decision Trees. Further optimization was done using hyperparameters. Finally, we have used the model(s) Accuracy, F1 Score, Precision and Recall to compare the results observed. The acquired result can be used further for the improvement of various antivirus systems in detecting ransomware softwares.

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who gave us the possibility to complete this project and report. We would like to show our gratitude to **Dr. C. Lakshmi**, Head of the department, software engineering, for giving us the opportunity to do the final year project. A special thanks to our final year project coordinators Dr. M. Uma and Mrs. J. Jeyasudha whose help, stimulating suggestions and encouragement helped us in coordinating the project and writing the report and our guide **Dr. Usha G.** who propelled the idea of this project and her full effort in guiding the team in achieving the goal as well as his encouragement to maintain our progress on track.

Ishan Chaudhari

Varshini V.

INDEX

I. INTRODUCTION

In today's world, computers are an essential part of our lives. For instance, you can write mail in a word processor, make changes to it, print multiple copies, send it to somebody via email halfway across the world in just a few clicks. All these tasks would have taken days to accomplish before, but now they are completed within moments. The fact of the matter is, whether it is a Small Business, or an Enterprise, they all rely on computer systems today. Pairing this with the rise in cloud computing, rather poor cloud security, and smart devices everywhere contributes to a multitude of threats.

Malware is a term used to define any piece of software built with the intention of harming the computer system, user or the data. These malwares were first introduced during the early 1970s to cause disruptions and as much damage as possible. Types of malwares include Viruses, Worms, Trojans and Spywares, furthermore in this research paper we will be focusing on detection of ransomwares.

Ransomware is a type of malign software, which when given access to a users computer system, will encrypt the data essentially rendering the system useless until the ransom amount is paid usually in the form of bitcoins. These softwares are able to set up their working environment in the background, also creating a backdoor to the system in the process to remain updated throughout the duration of attack. After the encryption is performed, the user is prompted with a ransom note (usually desktop wallpaper is set to the ransom note) with the details to the transaction ID and an intimidating note asking for money.

Ransomware's first cases date back to 2005 in Russia. Since then these types of malwares have spread across the world, advancing in method of implementation and targeting of their victims. Cryptolocker first surfaced in September 2013 and was advanced enough to target all versions of Windows. Victims would accidentally open these files and send them via email impersonating UPS, DHS etc. Once activated, the malware would get to work and prompt with a timer of 72 hours and a ransom.

In this paper, we have analysed some of the most popular ransomwares such as WannaCry, Locky, CryptoShield, Crysis, Win32.Blocker, Unlock26. These ransomware samples have been used from the ISOT Ransomware Dataset and contain trace files of the ransomwares.

We have used multiple machine learning algorithms and compared their results. The paper is divided into subsections as follows: Section Two will discuss related work on ransomware related research. The proposed methodology and implementation is discussed in the Fourth section. Section Five discusses the results observed. Finally, with Section Six we conclude our paper.

II. PROJECT OVERVIEW

1. LITERATURE REVIEW

There has been a surge in the use of machine learning approaches to detect and prevent ransomware attacks. A detailed study was made[2] in their paper that studied the detailed working of a ransomware attack and the different types of ransomware. They analyzed the features like CPU user usage, system usage, RAM usage, receive packet and byte, send packets, send bytes, receive packets, receive bytes, and netflows of the dataset. They used various machine learning algorithms such as KNN, Naïve Bayes, Random Forest, SGD, SVM, Logistic Regression, Bayesian Network to get desired output. They showed the comparison of all the accuracies and results as well. But every algorithm has its own advantages and disadvantages, so the best suited model which can be handled by the system is to be chosen. We improve on this study further by using different machine learning algorithms. We used the behaviour of the ransomware trace files to train and test the model.

A paper written by Ban Mohammed Khammas[3] is a static method of detection. They used a random forest classifier to detect and found out that a high accuracy can be achieved by changing the seed value to 1 and tree numbers 100. They also used Frequent Pattern Mining technique to directly extract the features from raw byte. This showed an increase in efficiency.

Next paper by Eduardo Berrueta, Daniel Morato, Eduardo Magaña, Mikel Izal[4]. They made an in depth survey that concentrates on various detection models. In comparison to the previous studies, they offer a survey on different ransomware families and propose a few detection algorithms.

In the paper[5] , they have used advanced data analytics techniques to find ransomware related transactions and bitcoin addresses. But using bitcoin technology has its own disadvantages of varying bitcoin transaction addresses.

In the paper[6] , they have compared the flexibility and stability of Machine Learning algorithms for security. A case study was done on evaluation of resilience using the generative adversarial network. They emphasized on the necessity of improving machine learning based approaches before final deployment. We gained a lot of insight on the various strategies and using ML techniques.

Lastly, the paper[7] helped us understand the current literature existing on the subject. Its survey consisted of research done in malware detection, prevention and recovery of the system after the attack. It helped us understand the existing work and further scope of research in this field from various papers in a summarized manner.

2. PROBLEM DESCRIPTION

In today's technologically advanced world, Cyber Security attacks are increasing day by day. Ransomware has become one of the most popular forms of attacks recently growing 350% in the year 2018. It is anticipated that every 14 seconds an enterprise falls to a ransomware attack and during the period between 2013 to 2016, the most common forms of ransomware variants were CryptoLockers and CryptoWalls.

The ransomware has become a serious threat and challenge for computers and technology especially for cryptocurrency and blockchain. It requires instantaneous attention to avoid further economic and ethical blackmail.

The project aims to tackle this issue by providing an improved and an efficient strategy to detect ransomware attacks by dynamically approaching the problem.

3. REQUIREMENTS GATHERING

- **Purpose:**
    - The purpose of this project is to detect and prevent ransomwares and find out which machine learning algorithms are suited best for the task at hand.
- **Product Functions:**
    - Detection of ransomware applications and preventing them from executing.
    - Using system traces like API calls, Network accesses, Directory scanning.
    - Determine which ML algorithm performs best.

4. REQUIREMENTS ANALYSIS

- **Functional Requirements:**
    - Detection of ransomware software.
    - Using system traces like API calls, Network accesses, Directory scanning

- Preprocessing of data.
- Training model on provided goodware and malicious dataset.
- Trained models must be able to differentiate between safe and malicious addresses.
- Compare results from all trained models.
- Determine the best trained model.
- Code must be implemented in Python.

- **System Requirements:**
  - Windows XP / Vista / 7 / 8.x / 10
  - 2 GB RAM
  - 10 GB storage
  - Working internet connection
  - Python 3.9.1 and above
  - Jupyter Notebook

- **Non - Functional Requirements:**
  - Accuracy - The output provided by the algorithms must be consistent and accurate.
  - Performance - The trained model must be able to perform the given task smoothly and quickly.
  - Stability - The application must be able to work under stressful environments with stability.
  - Security - The input datasets and outputs must be secure and integrity must be maintained to prevent tampering of the data.

5. DATA SOURCE

We used the ISOT Ransomware Detection Dataset from the ISOT Research Lab, University of Victoria. It is a behaviour data of a collection of ransomware and benign samples. They had been obtained from Virustotal under educational license alongside numerous samples from anti-malware groups. The dataset consists of a total of 669 ransomware samples from ransomware families that are most widely used. The size of the entire dataset on disk is 428 GB. Apart from the ransomware samples, the dataset also includes 103 benign most used windows applications.

The dataset contains information about the analysis task, duration analysis, various memory regions, established network connections, processes created by sample, system API calls during the initial analysis, arguments, return values, strings extracted from the binary file of the analysed sample, different operations on file system and Windows registry.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	info/adde	info/start	info/dura	info/ende	info/owne	info/score	info/id	info/categ	info/git/h	info/git/f	info/moni	info/packa	info/route	info/custo	info/mach	info/mach	info/mach	info/mach
2	1.53E+09	1.53E+09	1336	1.53E+09	null	7.6	1413	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
3	1.53E+09	1.53E+09	1336	1.53E+09	null	6.6	1414	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
4	1.53E+09	1.53E+09	643	1.53E+09	null	6	1415	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
5	1.53E+09	1.53E+09	657	1.53E+09	null	6	1416	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
6	1.53E+09	1.53E+09	1966	1.53E+09	null	8.2	1417	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
7	1.53E+09	1.53E+09	1290	1.53E+09	null	6.4	1418	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
8	1.53E+09	1.53E+09	55	1.53E+09	null	0.6	1419	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
9	1.53E+09	1.53E+09	755	1.53E+09	null	23.6	1420	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
10	1.53E+09	1.53E+09	2075	1.53E+09	null	5.4	1421	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
11	1.53E+09	1.53E+09	1334	1.53E+09	null	8.2	1422	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
12	1.53E+09	1.53E+09	1345	1.53E+09	null	7.4	1423	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
13	1.53E+09	1.53E+09	2016	1.53E+09	null	17.6	1425	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
14	1.53E+09	1.53E+09	1974	1.53E+09	null	6.6	1426	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
15	1.53E+09	1.53E+09	1332	1.53E+09	null	11.8	1427	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
16	1.53E+09	1.53E+09	629	1.53E+09	null	2.8	1428	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
17	1.53E+09	1.53E+09	1991	1.53E+09	null	3.6	1430	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
18	1.53E+09	1.53E+09	641	1.53E+09	null	6.2	1431	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
19	1.53E+09	1.53E+09	1934	1.53E+09	null	5.2	1433	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
20	1.53E+09	1.53E+09	2643	1.53E+09	null	9.6	1434	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
21	1.53E+09	1.53E+09	5067	1.53E+09	null	7.8	1435	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
22	1.53E+09	1.53E+09	3719	1.53E+09	null	13.4	1436	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
23	1.53E+09	1.53E+09	1067	1.53E+09	null	6.6	1437	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
24	1.53E+09	1.53E+09	5691	1.53E+09	null	9.4	1438	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
25	1.53E+09	1.53E+09	2990	1.53E+09	null	5.8	1439	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
26	1.53E+09	1.53E+09	3804	1.53E+09	null	7.8	1441	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
27	1.53E+09	1.53E+09	2059	1.53E+09	null	8	1442	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
28	1.53E+09	1.53E+09	1357	1.53E+09	null	13	1443	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
29	1.53E+09	1.53E+09	4078	1.53E+09	null	6	1444	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
30	1.53E+09	1.53E+09	2746	1.53E+09	null	6	1445	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
31	1.53E+09	1.53E+09	2030	1.53E+09	null	8.2	1447	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
32	1.53E+09	1.53E+09	1354	1.53E+09	null	7.2	1448	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
33	1.53E+09	1.53E+09	675	1.53E+09	null	7.2	1449	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo
34	1.53E+09	1.53E+09	732	1.53E+09	null	7.4	1626	file	5575a517	5575a517	e19c4b4b529be2e90		internet	null	stopped	cuckoo1	Win764	VirtualBo

FIG 2.5.1 : DATASET CSV FILE

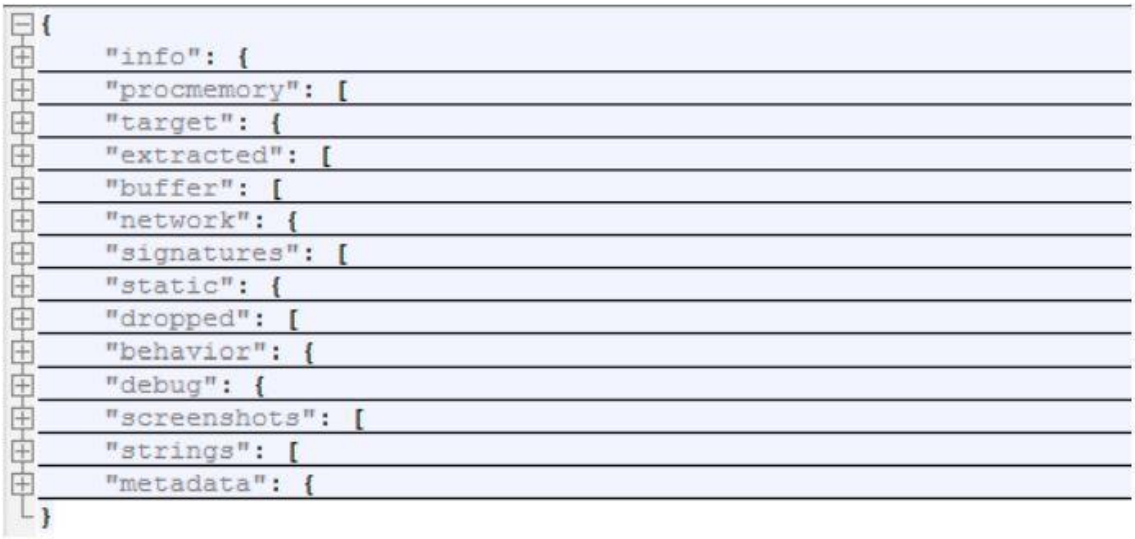

FIG 2.5.2 : DATASET DESCRIPTION

6. COST ESTIMATION

We are able to follow the heuristic method that is used for solving problems, studying, or discovery inside the sensible methods which might be used for attaining immediate goals. Those strategies are flexible and easy for taking quick selections through shortcuts and appropriate sufficient calculations, most possibly whilst running with complex records.

A regression model based on Lines of Code is known as Constructive Cost Model(COCOMO). A very reliable model in predicting time, effort, cost and quality. Our project comes under the basic Cocomo model, particularly organic type.

$$E = a_b \, KLOC^{\,b_b}$$

$$D = c_b \, E^{\,d_b}$$

where E is the effort applied in person-months, D is the development time in chronological months and KLOC is the estimated number of delivered lines of code for the project (express in thousands). The coefficients a_b and c_b and the exponents b_b and d_b are given in Table below

Software Project	a_b	b_b	c_b	d_b
organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
embedded	3.6	1.20	2.5	0.32

TABLE 2 : COCOMO MODEL

7. PROJECT SCHEDULE

Time period	Checkpoints of the project
23/10/2020 – 25/11/2020	Research for a topic of the project
26/10/2020 – 29/11/2020	Work on the problem statement and overview of the project
09-12-2020	Zeroth Review
10/12/2020 – 3/01/2021	Study about the project
4/01/2021 – 9/01/2021	Literature Survey
10/01/2021 – 14/01/2021	Requirement gathering and analysis
15/01/2021 – 19/02/2021	Project design, sequence and methodologies

20/02/2021 – 23/02/2021	UI designed and system architecture
24/02/2021 – 25/02/2021	Risk analysis and cost estimation
01/03/2021	First Review
02/03/2021 – 10/03/2021	Planned properly for the implementation of the project
11/03/2021 – 15/03/2021	Collected datasets
16/03/2021 – 25/03/2021	Module wise implementation
26/03/2020 – 27/03/2021	Debugging the program and testing simultaneously
28/03/2020 – 29/03/2021	Manual Unit testing (created test cases on test scenarios and tested each module individually)
30/03/2020 – 31/03/2021	Research Paper
01/04/2021	Review 2
02/04/2021 – 07/04/2021	Improving the UI a bit, implementing the module left, implementation of the module left
08/04/2021 – 13/04/2021	Integration testing and user testing
14/04/2021 – 20/04/2021	Performance Analysis

21/04/2021 – 30/04/2021	Result analysis and performance graph plotted
04/05/2021	Review 3

TABLE 3 : PROJECT SCHEDULE

8. RISK ANALYSIS

1. False Positives might be detected.
2. User Data Loss in case of failure to detect a malicious file.
3. Ransomwares not using bitcoin transactions could bypass the system.
4. Dataset could be insufficient for model training.
5. The metric selection process might not be sufficient for the team to identify what kind of ML solutions could be used.

9. Software Requirement Specification (SRS)

1. Introduction

a. Purpose

The primary purpose of the project is to improve detection techniques used by antiviruses using ML applications. The scope of the SRS is to define requirements and the modules to be used by the software.

b. Document Conventions

This document uses Times new roman as font theme for normal content. Font size of content is 12 while subheadings have a font size of 14 and they are in bold. In addition, important text in the content is highlighted by bolding the text.

c. Intended Audience and Reading Suggestion

The different types of reader that the document is intended for are software development teams, project managers, documentation writers, users, marketing

teams, and testers. This document contains a detailed description of the product including its functions, design, user documentation assumptions and interfaces as well. Readers are advised to read the document in the given order only to get a good understanding of the product. They should start from description and go in the given order. This document also contains system features and other non-functional requirements.

d. Product Scope

The scope of the project is to detect the ransomware application before it is able to do any significant damage to the user's system. It aims to improve existing detection systems using Machine Learning.

2. Overall Description

a. Product Perspective

- The project aims at providing developers of antivirus systems with an improved way to detect malicious files.
- It will help the antiviruses in classifying files.

b. Product Function

- Trained models must be able to differentiate between safe and malicious addresses.
- Detection of ransomware software.
- Using system traces like API calls, Network accesses, Directory scanning.
- Compare results from all trained models.

c. User Classes and Characteristics

The major users of the software are the Antivirus customers and other enterprises protecting their data from ransomware attacks. The developer too has the access of all the software to load the dataset and for maintenance purposes.

d. Operating Environment

- The algorithm used to train the dataset is coded in python language.
- The libraries used are NumPy, pandas, sklearn.
- Files are converted using DataFileConverter.
- Further Matplotlib is used for plotting graphs.

e. Design and Implementation Constraints

- The algorithm's complexity and time taken increases with larger dataset and more virus specimens.
- The system has to produce minimum false positives for smooth functioning of the user's system.
- High cost of deployment of the product.

f. User Documentation

All the specified user manuals with the working tutorials and on help maintenance will be taken care. The documents will all be in the IEEE format and will be distributed after the ISO certification.

g. Assumptions and Dependency

- Using Trace files of the virus to train the model using machine learning algorithms.
- Using the right machine learning algorithm for training the model.

3. External Interface Requirement

a. User Interfaces

- Front end: Python3

b. Hardware Interfaces

- Processor: Intel Core i7-7500U CPU
- Hard disk: 250 GigaBytes or Higher
- RAM: 8 GigaBytes

c. Software Interfaces

- Operating System: Windows 10
- Languages used: Python
- Tools: Jupyter notebook, anaconda3

4. System Features

The main services and functional requirements for the product may be illustrated via system functions.

a. Collecting Dataset

Creating a ransomware dataset for the training of the models is the primary task. Datasets must contain accurate values from authentic sources.

b. Training Models

Multiple algorithm models should be used with the dataset and should be trained and validated.

c. Validation

Once the model is trained with the training dataset, the model must predict the malicious files.

5. Other Non-Functional Requirement

a. Performance Requirement

The trained model must be able to perform the given task smoothly and quickly.

b. Safety Requirement

The input datasets and outputs must be secure and integrity must be maintained to prevent tampering of the data.

c. Software Quality Attributes

- Availability: the antivirus must be available 24*7 hours to the user.
- Correctness: the dataset used shouldn't contain false values.
- Usability: The software should be understandable to users

d. Business Rules

- The antivirus will focus on ransomware prevention therefore protecting the users.
- It will be provided at a monthly subscription basis being profitable for the company.

III. ARCHITECTURE AND DESIGN

1. SYSTEM ARCHITECTURE

System Architecture is a model that describes the structure and behavior of a complex system. It comprises all the components and the overview of the whole system.

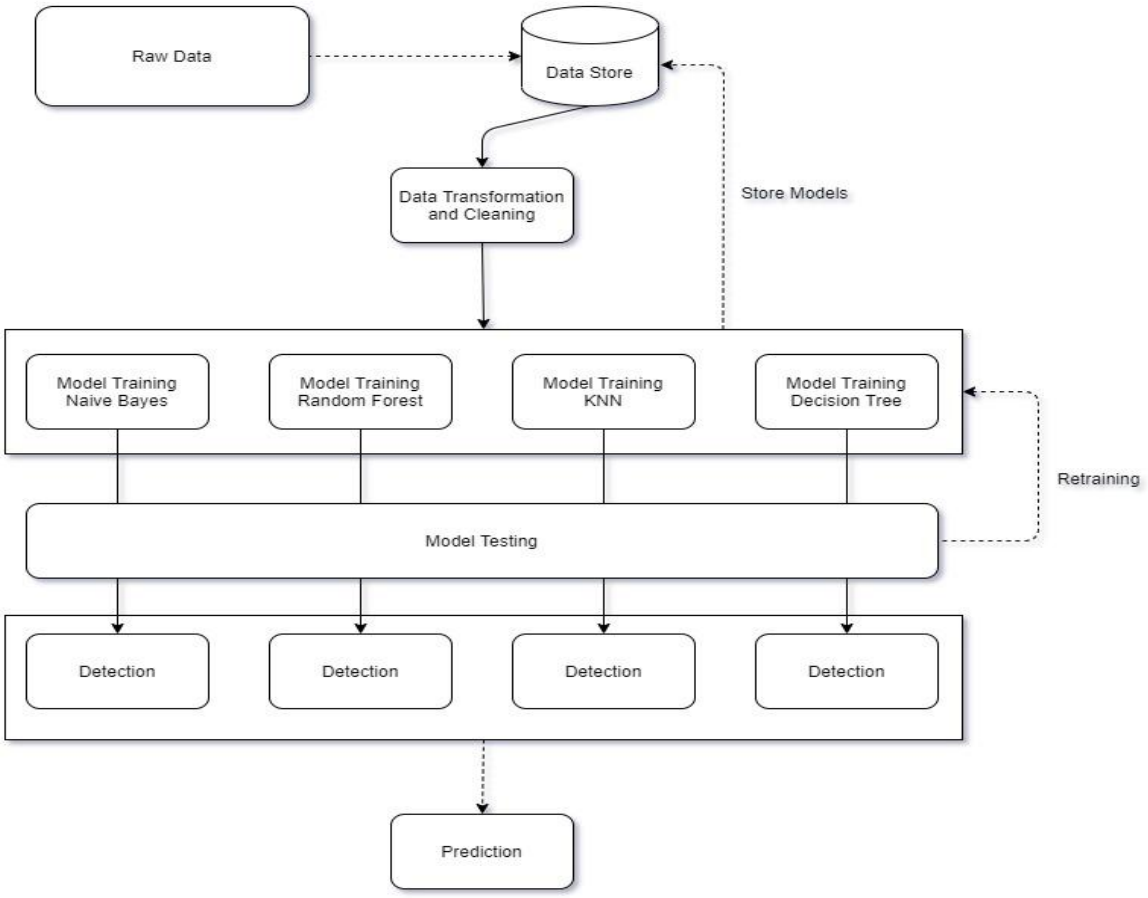

FIG 3.1 : SYSTEM ARCHITECTURE

- Raw Data : This consists of unorganized raw data. It contains trace files from existing ransomwares about their behavioural data.
- Data Processing : This module organizes and cleans the raw data to extract usable data for model training and testing. It will organize all the benign files and infected files together and create two sets for training and testing.

- Model Training : This module contains four models that will utilize the processed data fed in from Data Processing and train the respective models. The training will be done over iterations and will be tweaked to improve the results.
- Testing : This module will validate the results and compare them to the trained models.

2. INTERFACE PROTOTYPING

A User Interface allows the user to interact with the software effortlessly. By presenting graphical images that can be interacted with, the user can understand the interface without any issues and allows for a wider range of usability.

FIG 3.2 : INTERFACE PROTOTYPE

3. DATA FLOW DIAGRAM

DFD is a diagrammatic representation of functions/ jobs that manipulate, catch, store, and distribute knowledge between a system and its elements. It is a simple visual demonstration which makes it a means of interaction between the User and System designer.

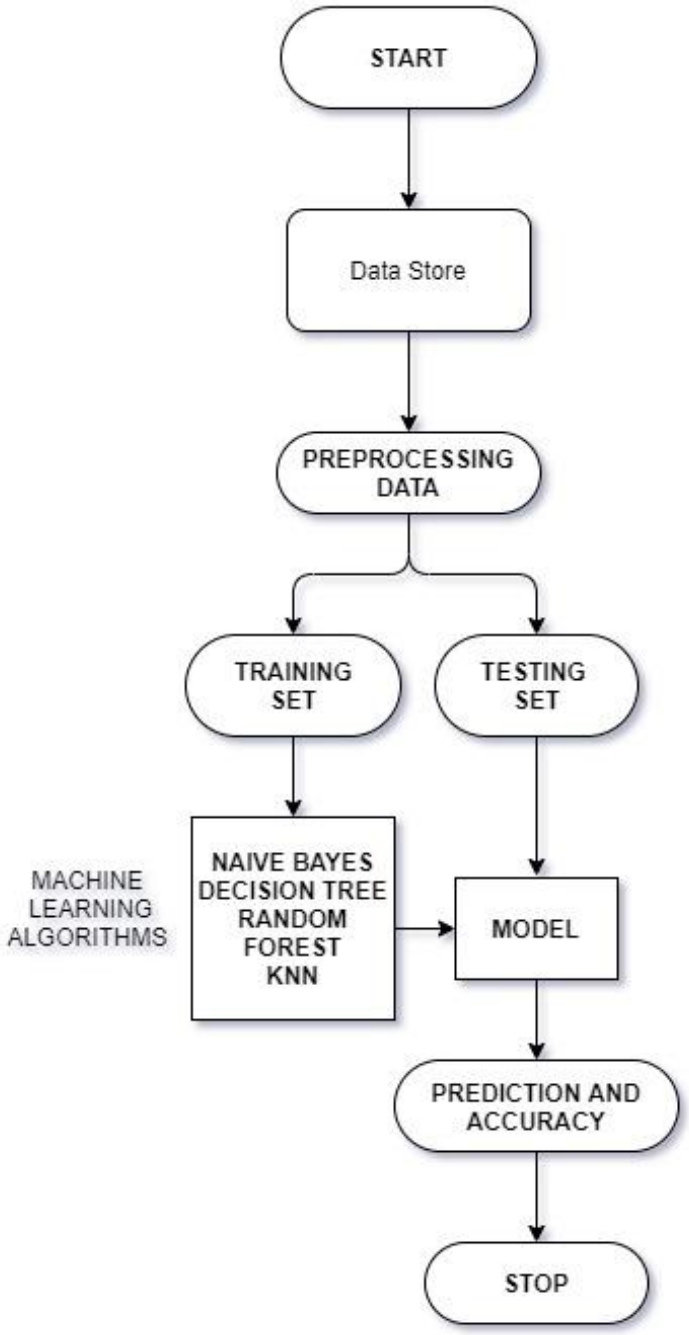

FIG 3.3 : DATA FLOW DIAGRAM

4. USE CASE DIAGRAM

A use case diagram shows the user's interaction with the system. It depicts relationships among the user and the special use cases.

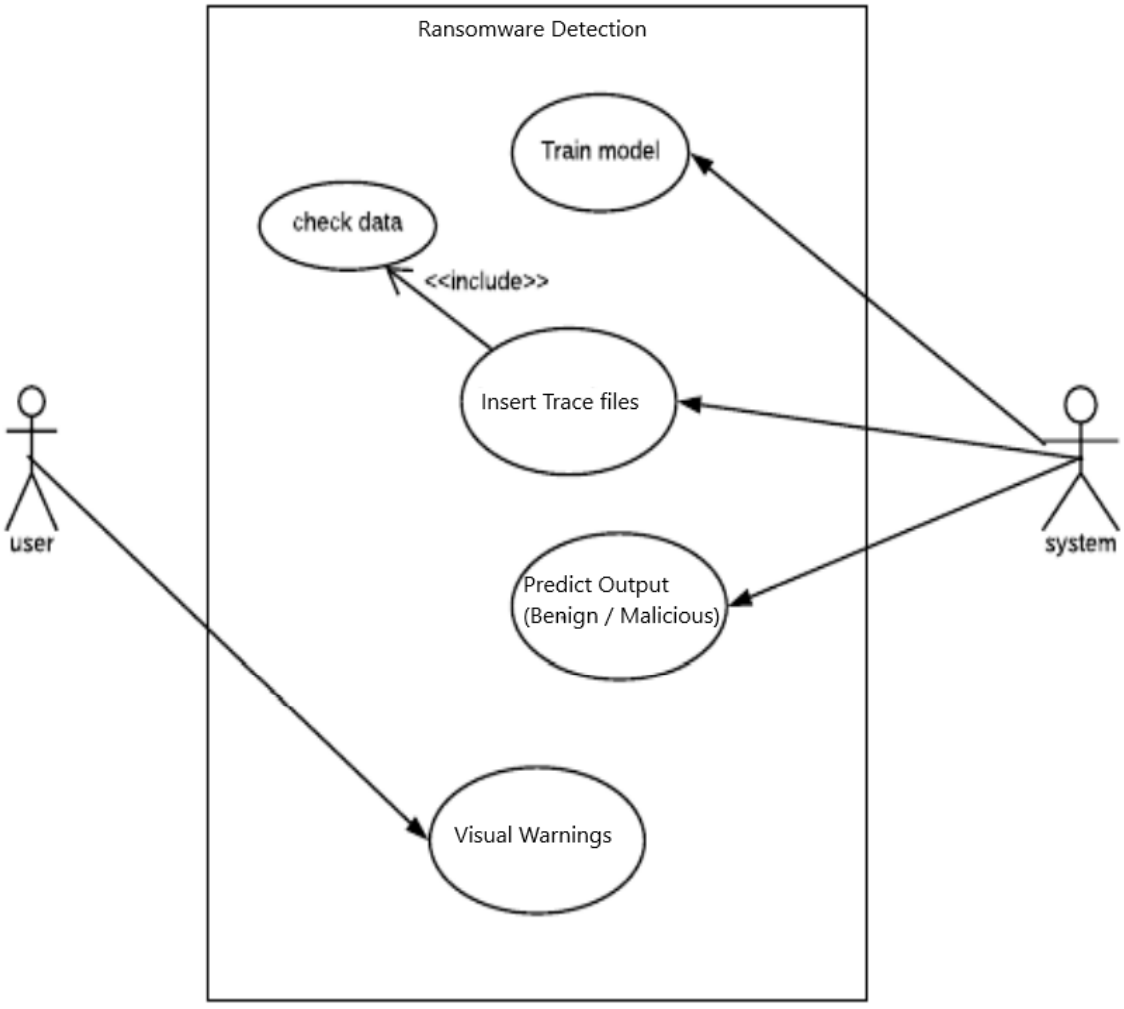

FIG 3.4 : USE CASE DIAGRAM

5. SEQUENCE DIAGRAM

The sequence diagrams, additionally referred to as interaction diagrams, show the interaction between the items with time sequence. It shows the objects and classes involved in the case. It also shows the message between them.

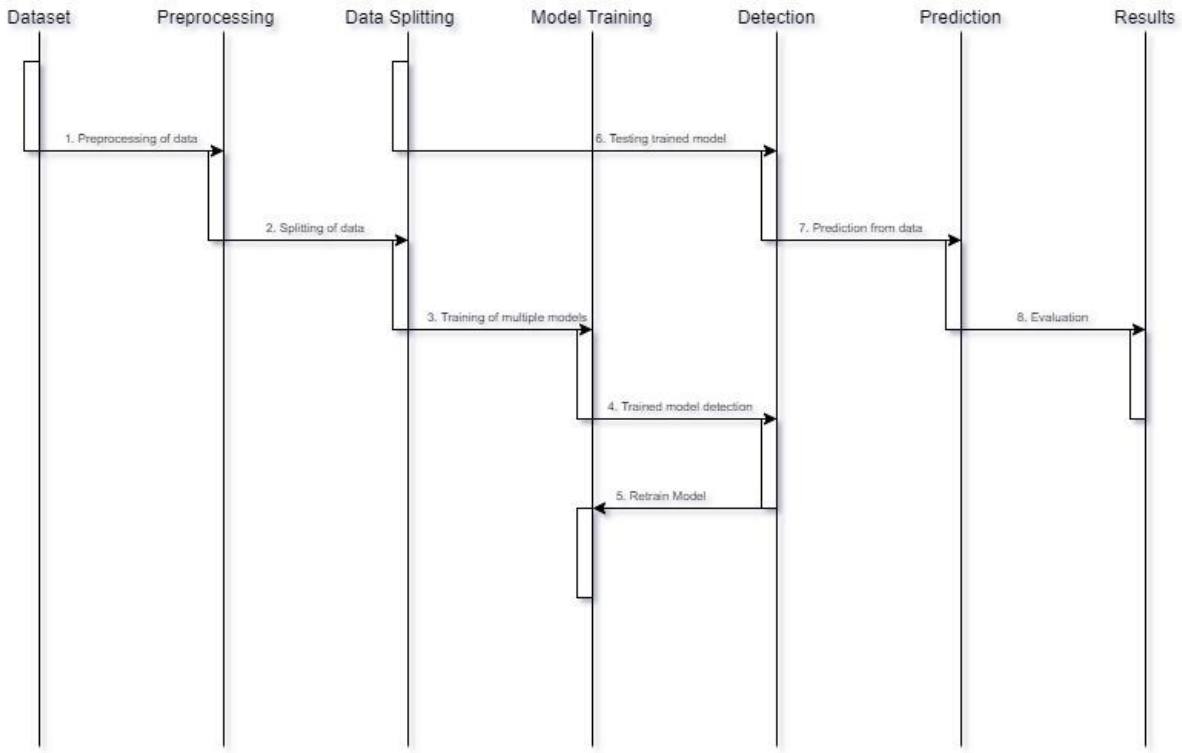

FIG 3.5 : SEQUENCE DIAGRAM'

6. CLASS DIAGRAM

The class diagram is a type of UML diagram which describes the system structure by depicting classes. It also shows their attributes, operations and visibility.

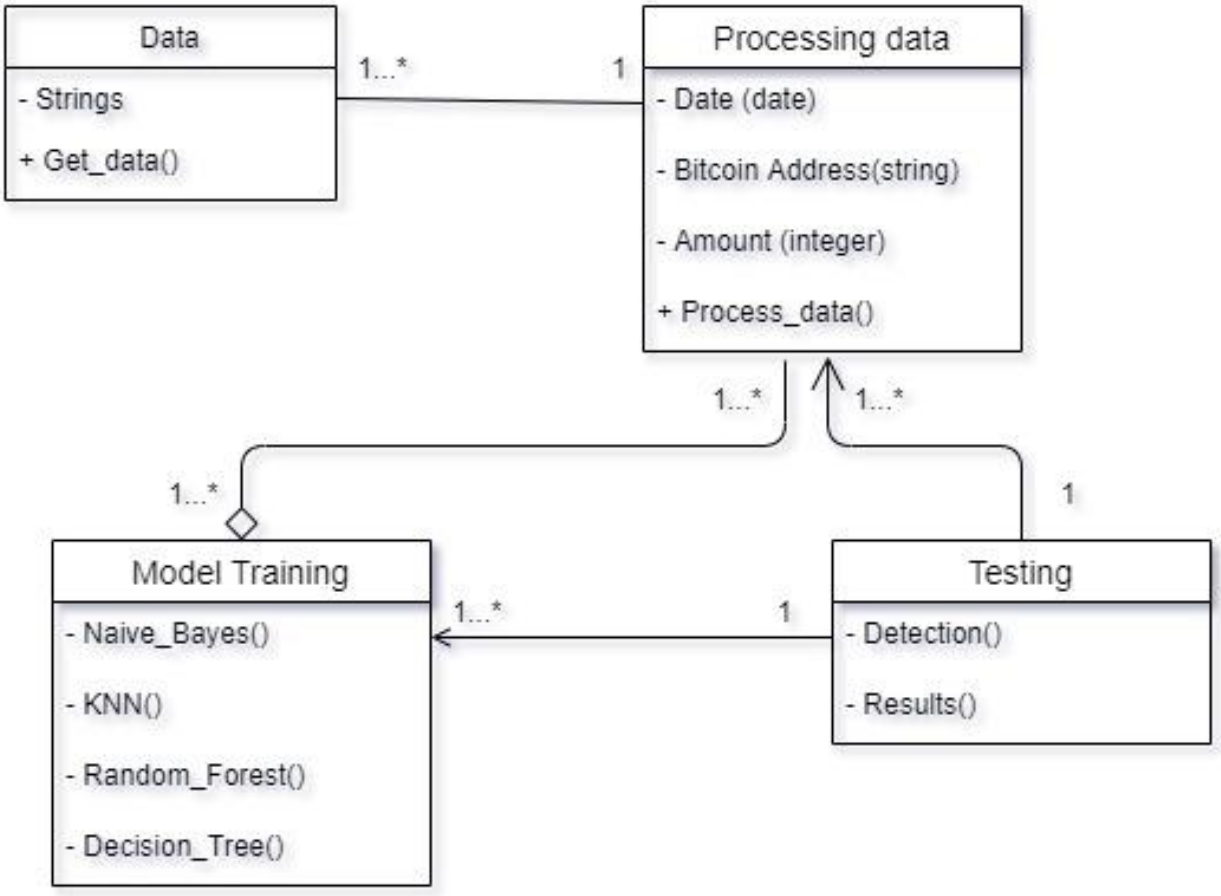

FIG 3.6 : CLASS DIAGRAM

7. ACTIVITY DIAGRAM

An activity diagram shows the behaviour of a system. It portrays the flow of control from a begin factor to the finishing point displaying the diverse choice paths that are existing while the activity executes.

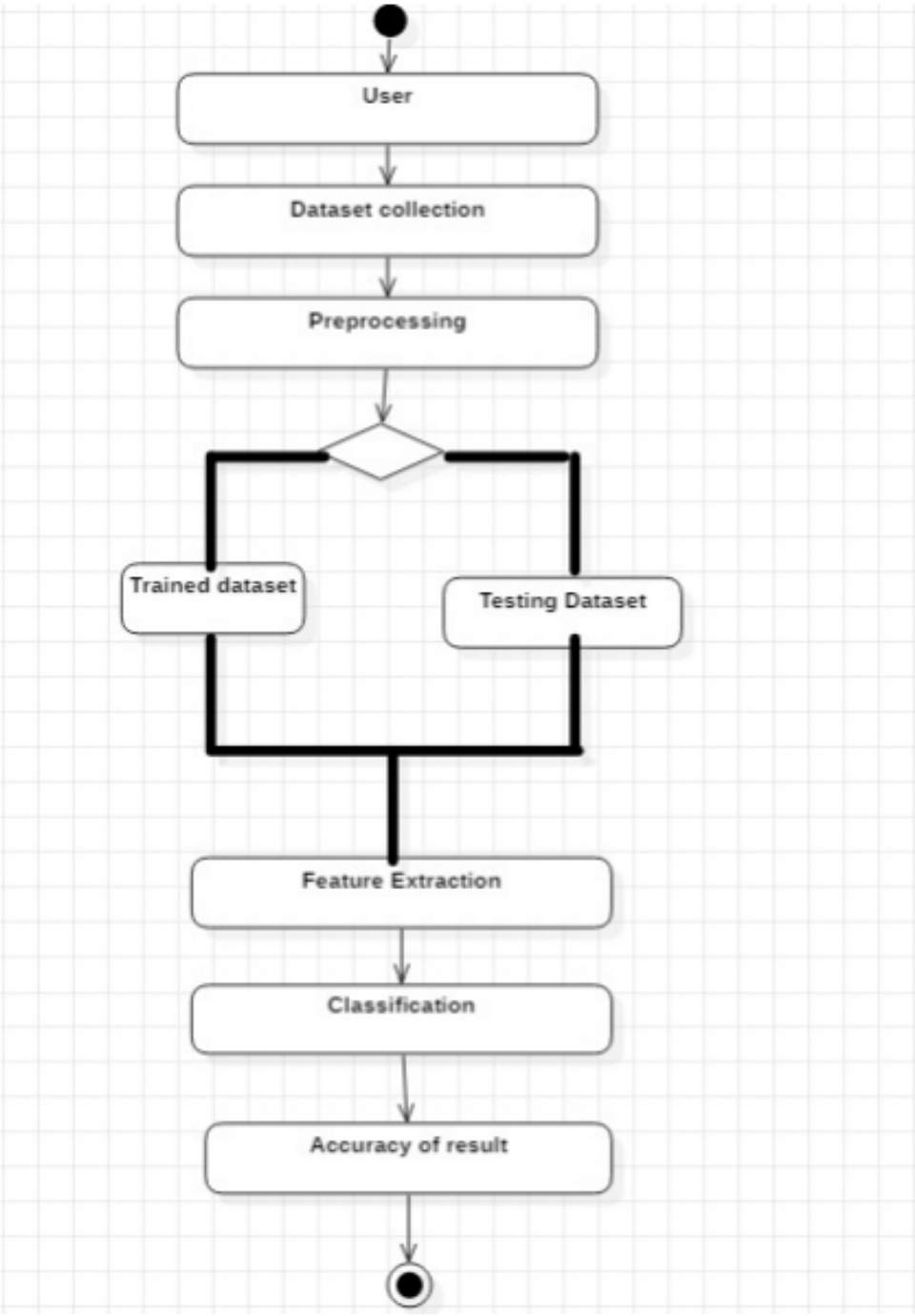

FIG 3.7 : ACTIVITY DIAGRAM

8. DEPLOYMENT DIAGRAM

A deployment diagram will show the execution architecture of a system, it contains the system hardware and software nodes and the middleware connecting them.

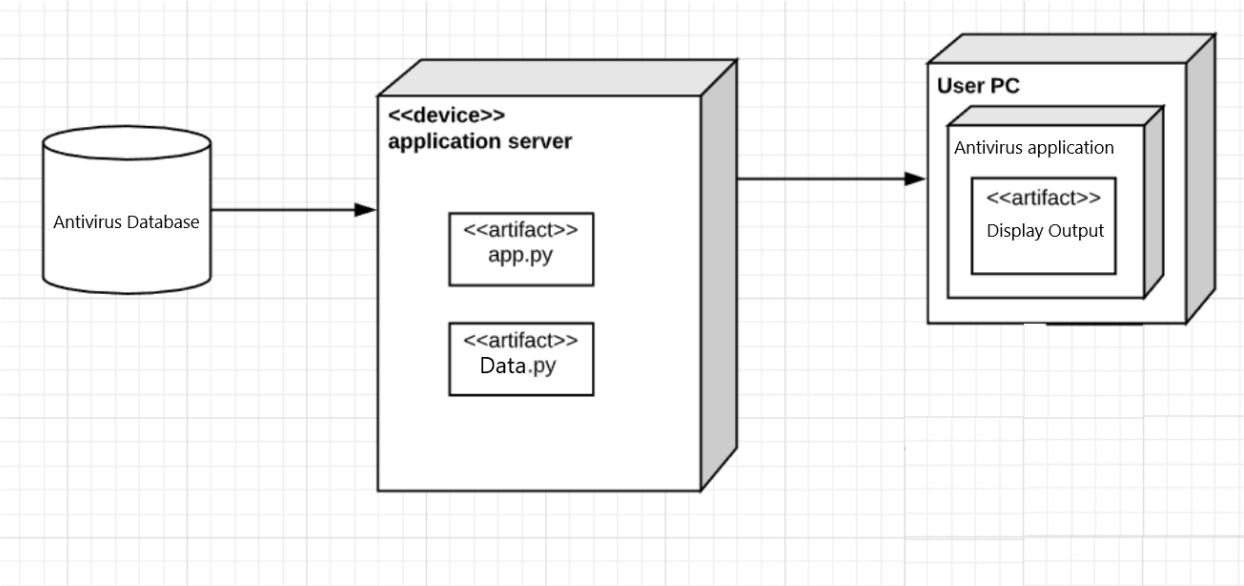

FIG 3.8 : DEPLOYMENT DIAGRAM

IV. IMPLEMENTATION

1. DATABASE DESIGN

a. ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) Diagram is a conceptual database model in the form of a flow chart which shows how different entities of a system are related to one another, using specific relationships which can exist between instances.

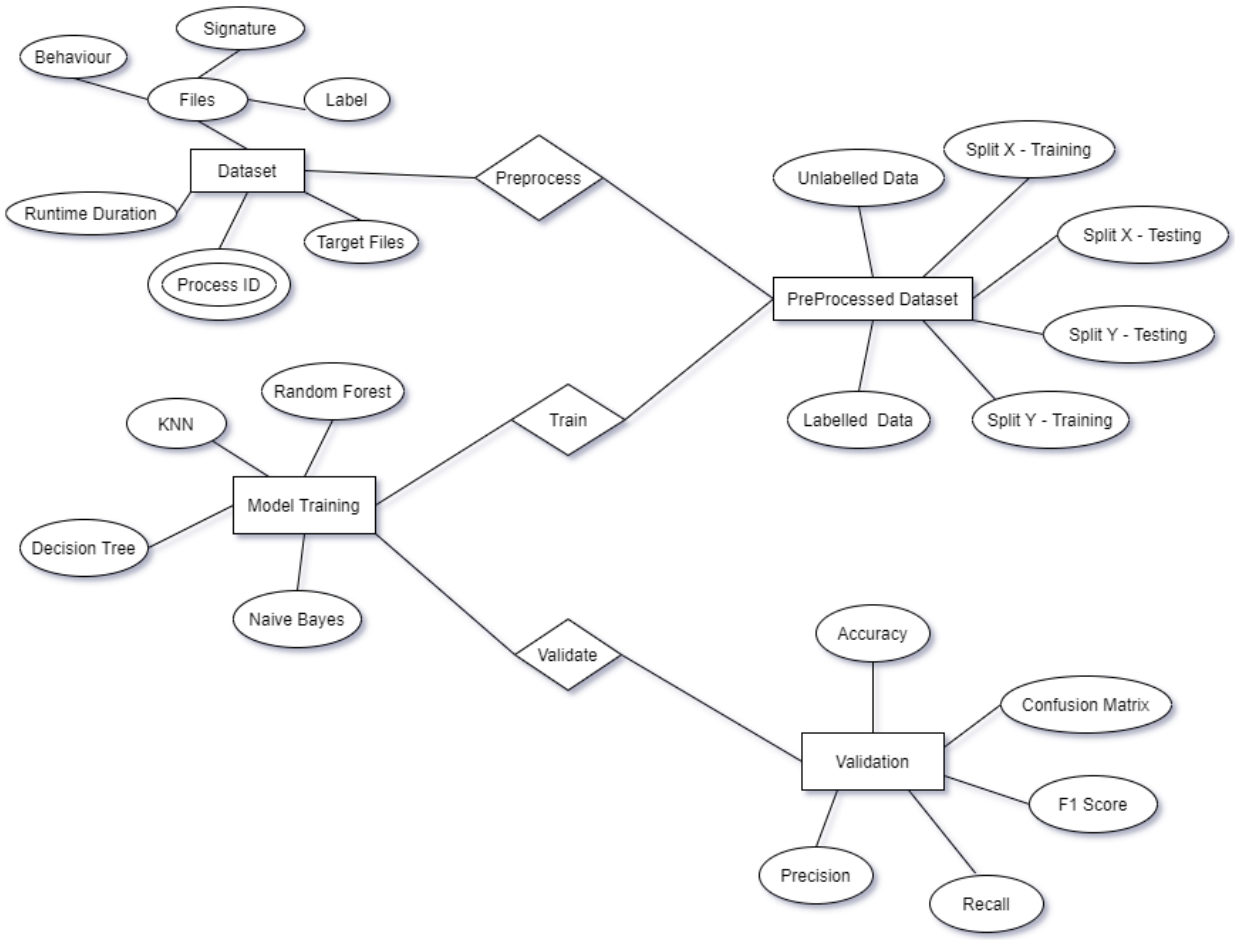

FIG 4.1.1 : ENTITY RELATIONSHIP DIAGRAM

b. RELATIONAL DIAGRAM

The relational model is an approach to managing information the usage of a structure and language constant with first-order predicate common sense, in which all data is represented in terms of tuples, grouped into relations.

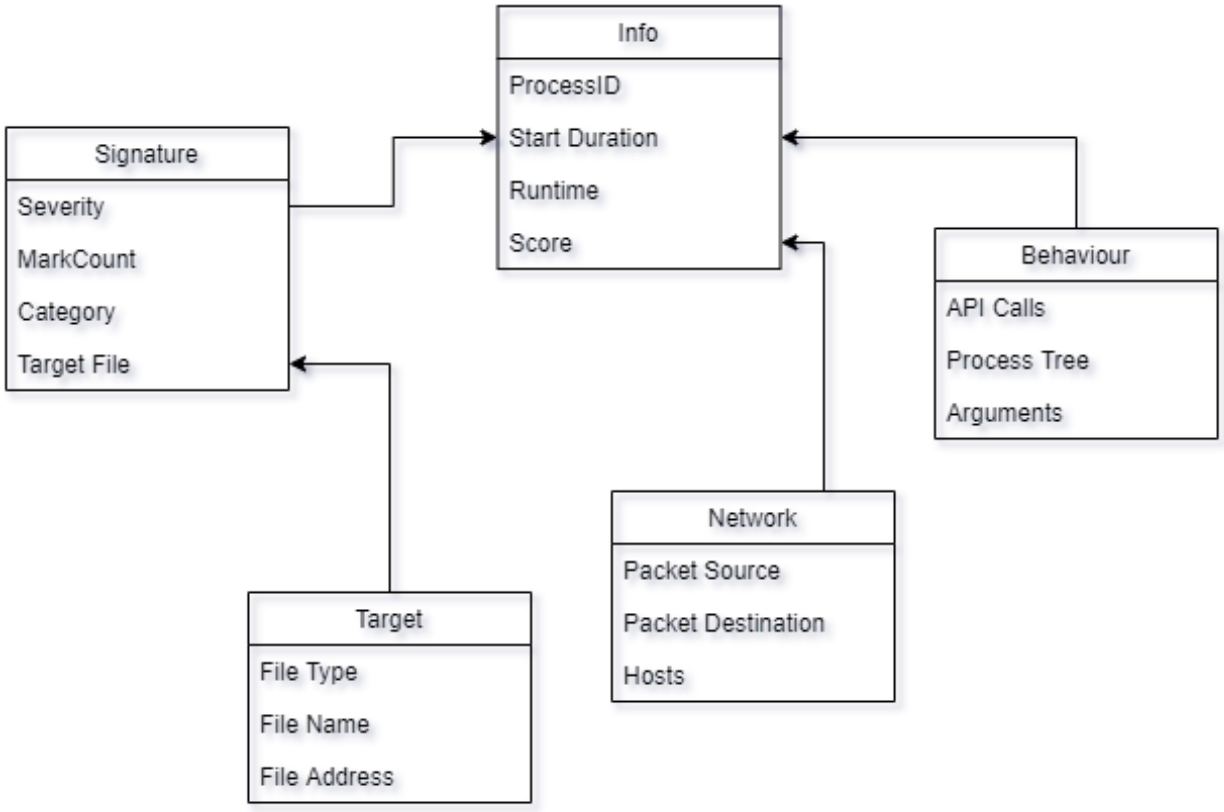

FIG 4.1.2 : RELATIONAL DIAGRAM

2. USER INTERFACE

Interface here provides the user with easy access to all the tools provided by the application. The user can immediately scan for threats, update the existing database for newer malwares, perform selected file scans.

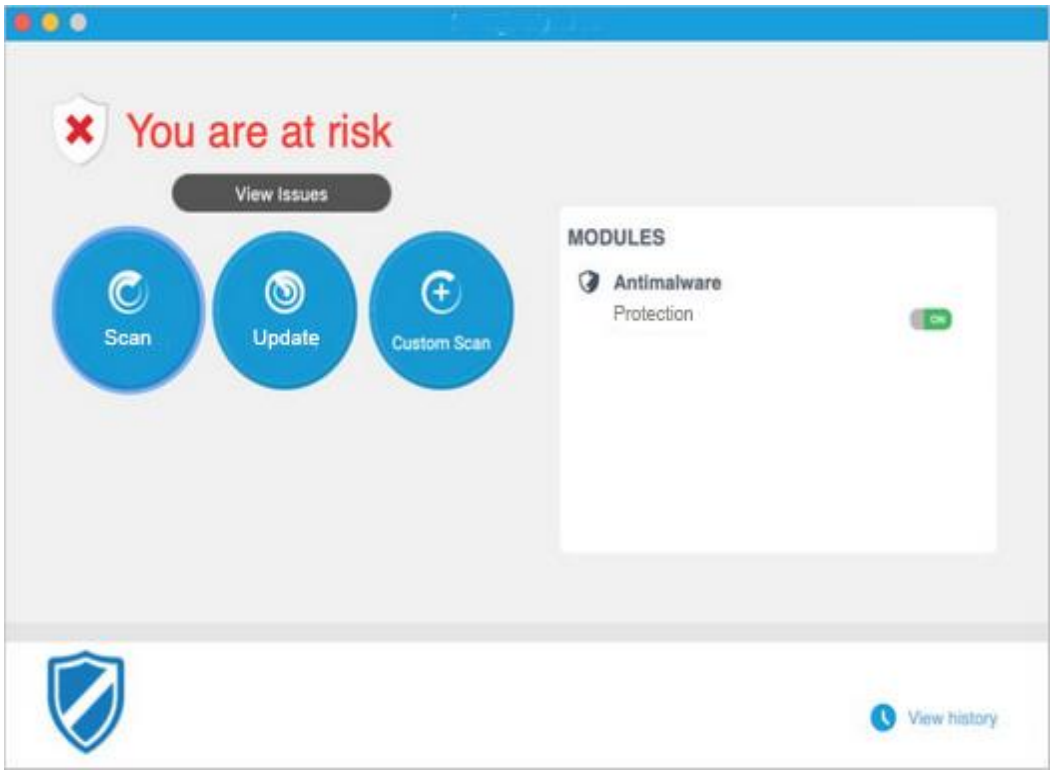

FIG 4.2 : USER INTERFACE

3. MIDDLEWARE

For implementation, we have used Jupyter Notebook and Anaconda as it allows for an excellent workflow.

The code was written in Python. Some of the libraries used in the code were Pandas, Numpy, SKLearn, MatPlotLib. Various dataset loading issues were solved by encoding the data using label encoding. The programming language used in our project is Python. It is a high-level programming language which uses an interpreter. Python can be dynamical and supports garbage collection.

The python IDE used is Spyder. Spyder is a powerful environment written in Python language. It is used by engineers, data analysts and scientists. It provides a combination of analysis, enhanced editing, debugging and profiling functionality of a comprehensive development tool..

We used scikit.learn to implement ML in python. It is a library used for ML applications which features different types of clustering, classification, and regression algorithms including SVM, k-means, random forests and Naive Bayes.

The entire research project was implemented in Windows 10 64bit.

4. CODE

In [ ]:
```
#Loading Dataset

import pandas as pd
import numpy as np
data = pd.read_csv('final datasetcleaned.csv')
```

In [ ]:
```
#Display initial information
data.shape
data.describe(include = 'all')
```

In [ ]:
```
#Labeled Encoding
from numpy import asarray
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
data['signatures/markcountENC'] = labelencoder.fit_transform(data['signatures/markcount'])

data['signatures/description'].astype('category')
data['signatures/descriptionENC'] = labelencoder.fit_transform(data['signatures/description'])
data['signatures/severity'].astype('category')
data['signatures/severityENC'] = labelencoder.fit_transform(data['signatures/severity'])
data['signatures/marks/call/category'].astype('category')
data['signatures/marks/call/categoryENC'] = labelencoder.fit_transform(data['signatures/marks/call/category'])
data['signatures/marks/call/api'].astype('category')
data['signatures/marks/call/apiENC'] = labelencoder.fit_transform(data['signatures/marks/call/api'])
data['signatures/marks/type'].astype('category')
data['signatures/marks/typeENC'] = labelencoder.fit_transform(data['signatures/marks/type'])
data['signatures/nameENC'] = labelencoder.fit_transform(data['signatures/name'].astype('str'))
data['target/file/typeENC'] = labelencoder.fit_transform(data['target/file/type'].astype('str'))
df = data
df.describe()
```

In [ ]:
```
#Preprocessing & Cleaning
#Removing NaN value columns

dataset_unlabeled = df.apply (pd.to_numeric, errors='coerce')
del dataset_unlabeled['signatures/description']
del dataset_unlabeled['signatures/markcount']
del dataset_unlabeled['signatures/severity']
del dataset_unlabeled['signatures/marks/call/category']
del dataset_unlabeled['signatures/marks/call/api']
del dataset_unlabeled['signatures/marks/type']
del dataset_unlabeled['signatures/name']
del dataset_unlabeled['target/file/type']

#Saving DataFrame to CSV file
df.to_csv(r'C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Dataset_unlabeled.csv', index = False)
```

In [ ]:
```
#Plotting Graphs
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot

dataset_unlabeled = pd.read_csv('dataset_unlabeled.csv')
dataset_labeled = pd.read_csv('dataset_labeled.csv')

# box and whisker plots
dataset_labeled.plot(kind='box', subplots=True, layout=(1,15), sharex=False, sharey=False, figsize = (40,20))
pyplot.savefig('bandw.jpg', dpi=150)
pyplot.show()
```

FIG 4.4.1 : CODE

```
# histograms
dataset_labeled.hist(figsize = (20,20))
pyplot.savefig('hist.jpg', dpi=150)
pyplot.show()

# scatter plot matrix
scatter_matrix(dataset_labeled, figsize = (50,50))
pyplot.savefig('scatter.jpg', dpi=150)
pyplot.show()
```

In [ ]:
```
import pandas
import matplotlib
import matplotlib.pyplot as plt

dataset_unlabeled = pd.read_csv('dataset_unlabeled.csv')
dataset_labeled = pd.read_csv('dataset_labeled.csv')

dataset_labeled['label'].value_counts()
pyplot.clf()
pyplot.figure(figsize=(12, 8))
params = {'axes.titlesize':'18',
          'xtick.labelsize':'10',
          'ytick.labelsize':'10'}
matplotlib.rcParams.update(params)
pyplot.title('Benign and Infected files')
#df.plot(kind='barh')
dataset_labeled['label'].value_counts().apply(np.log).plot(kind='barh')

pyplot.show()
```

In [ ]:
```
#DataSplitting

from sklearn.model_selection import train_test_split

data = dataset_labeled.drop('label', axis=1)
label = pd.DataFrame(dataset_labeled['label'])
x_train, x_test, y_train, y_test = train_test_split(data, label,test_size=0.25)
dataset_labeled.to_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\clean_labeled.csv",
          index=False)
x_train.to_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_data.csv",
           index=False)
y_train.to_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_label.csv",
           index=False)
x_test.to_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\test_data.csv",
           index=False)
y_test.to_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\test_label.csv",
           index=False)
```

In [ ]:
```
#Load Training & Testing Datasets

x_train = pd.read_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_data.csv")
y_train = pd.read_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_label.csv")
x_test = pd.read_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_data.csv")
y_test = pd.read_csv(r"C:\Users\Ishan Chaudhari\Desktop\Security Research\Project\Testing\train_label.csv")
```

In [ ]:
```

# comparing algorithms and training models
from pandas import read_csv
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
```

FIG 4.4.2 : CODE

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from  sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
# Spot Check Algorithms
models = []
models.append(('KNN', KNeighborsClassifier(n_neighbors=19)))
models.append(('Decision Tree', DecisionTreeClassifier(min_samples_leaf=30)))
models.append(('Naive Bayes', GaussianNB()))
models.append(('Random Forest', RandomForestClassifier(n_estimators=80, max_depth=3,random_state=0,min_samples_leaf=10))


# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=3, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, x_train, y_train.values.ravel(), cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
# Compare Algorithms
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

In [ ]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [ ]:
```
model = KNeighborsClassifier(n_neighbors=19)
model.fit(x_train, y_train.values.ravel())
prediction = model.predict(x_test)
print("KNN")
print("Accuracy = ",accuracy_score(y_test, prediction))
print("Confusion Matrix =")
print(confusion_matrix(y_test, prediction))
print(classification_report(y_test, prediction))
```

In [ ]:
```
model = DecisionTreeClassifier(min_samples_leaf=30)
model.fit(x_train, y_train.values.ravel())
prediction = model.predict(x_test)
print("Decision Tree")
print("Accuracy = ",accuracy_score(y_test, prediction))
print("Confusion Matrix =")
print(confusion_matrix(y_test, prediction))
print(classification_report(y_test, prediction))
```

In [ ]:
```
model = GaussianNB()
model.fit(x_train, y_train.values.ravel())
prediction = model.predict(x_test)
print("Naive Bayes")
print("Accuracy = ",accuracy_score(y_test, prediction))
print("Confusion Matrix =")
print(confusion_matrix(y_test, prediction))
print(classification_report(y_test, prediction))
```

FIG 4.4.3 : CODE

V. VERIFICATION AND VALIDATION

The proposed system has to be tested to show its validity. Testing is considered the least innovative step in the whole software development process. It is the process, in the true sense, that can pull out the ability of the other phases that lets it work. Verification and validation is the method of evaluating whether a software program satisfies requirements and expectations, which fulfills the expected task. Verification is the method of ensuring that a program is meeting its purpose without any glitches. It's the method of determining whether or not the product being produced is correct. This verifies that the substance being produced satisfies the specifications we have. Validation is the method of testing whether the output of the algorithm is up to the mark. It is the method of testing product validity, that is, it tests the correct quality for what we are producing.

1. UNIT TESTING

This is a software test stage, wherein specific units / elements of applications are evaluated. The purpose is to verify that each application system is working as planned. A unit is the tiniest part of all testable applications. Usually, it consists of 3 different phases: planning, scripts and cases, and unit testing. It is prepared and re-examined in the first phase. The next stage is to do the test cases and scripts then check the application. In this project, the individual modules like data pre-processing, clustering, user registration and login are divided into small basic units to test its functionalities and make sure that it is working as expected.

Test Id	**Test Actions**	**Input**	**Expected Output**	**Actual Output**	**Pass/ Fail**
T1	Check the integrity of the dataset	Access the dataset and check the attributes of the slices	Access to the dataset and all its attributes	Success	Pass
T2	Check the data encoding	Load dataset and display dataset in Jupyter Notebook	No NaN values	Success	Pass
T3	Resizing the Features of dataset and checking dataset integrity	Trace data from Application	Resized the features of the dataset	Success	Pass
T4	Splitting and saving Preprocessed data to training and testing sets	Trace data from application	Pre-processed data file at the destination location with a counter to keep track	Success	Pass
T5	Preprocessing data further using Cross Validation and checking integrity	Pre-processed data file	Improvement in model training	Success	Pass

T6	Training the data	Pre-processed data file	Detection of malicious files from a mix of benign files and infected files	Success	Pass

TABLE 4 : UNIT TESTING

2. INTEGRATION TESTING

This is the next phase of software tests in which two or more modules of a system are integrated and tested together. It is done to either test the integration between components of a system or between different parts of a system. In this project, it is done manually.

Test ID	Test Objective	Test Description	Expected Result
T1	Check the link between converting json files to CSV files	Click on the convert data to convert individual ransomware json files to combined csv file	Data must be appended to a csv while containing all trace files from each ransomware in dataset
T2	Check the integration between importing the data in CSV file with the algorithm	Python library will read the data from the CSV file and apply it to the machine learning algorithm	Data in CSV file should be imported successfully
T3	Check the integration between training the algorithm with model data and predicting the future price of ransomware	The algorithm will be trained using a python library called sklearn with Decision tree, random forest, KNN and naive bayes	Algorithm trained successfully
T4	Check the integration between predicting the price and converting the predicted data in the form of tables and graphs	The predicted data will be depicted using the python library like matplotlib	Data represented successfully using graphs and tables

TABLE 5 : INTEGRATION TESTING

3. USER TESTING

User testing is the mechanism by which actual users who conduct particular activities in practical circumstances test the functionality and features of a website, device, software, or service.

Test ID	Test Action	Input	Expected Output	Actual Output	Pass / Fail
T1	User runs Training module	User clicks Run button	Model training begins and model accuracies are shown	Model training begins and model accuracies are shown	pass
T2	User runs testing module	User clicks run button	Trained model is checked and results are shown	Trained model is checked and results are shown	pass

TABLE 6 : USER TESTING

4. SIZE – LOC

The whole project has been coded using python programming language on a dynamic web project. using many libraries and datasets. The front-end of the system displays information related to the top gainers and the top losers. It also gives information about the future prediction of each crop independently on individual web pages. This information is preprocessed and stored in datasets having format CSV using data cleaning Algorithm. The python source code processes all the modules with the different datasets.

The project size for this recommendation system is: 1.150 The cost estimation effort estimation depends on the size of the project. The size of the project is the number of lines of code.

5. COST ANALYSIS

Our proposed system is an organic software project using the basic COCOMO model because team size is small. The calculations are as follows-

So considering we may have 350 lines of code;

$E = a_b \text{ KLOC }^{b}_{b}$

E=(2.4)(350/1000)(1.05)

E= 0.882

D=(2.5)(0.882)(0.38)

D= 0.837

N=E/D

= 0.882/0.837

N = 1.05 persons

And accordingly, we can plan to use 2 people to finish it in less than the duration of the project.

- HARDWARE NEEDED

    1. Processor. : Pentium Dual Core 2.3 GHz
    2. Hard Disk. : 250 GB or Higher
    3. RAM : 4GB

  Approx minimum cost – Rs 35000

- SOFTWARE NEEDED

    1. OS . : Windows
    2. Language used. : Python
    3. Tools. : Spyder

  Approx minimum cost – Rs 0

6. DEFECT ANALYSIS

Defect Analysis is a technique used to identify and prevent the problems from occurring and ensure continuous quality improvement in the system. It can be done using Pareto Chart or Cause-Effect Analysis (using Fishbone Diagram). We have used Cause-Effect Analysis for our project as it will help improve the quality and detect the defects in an efficient way.

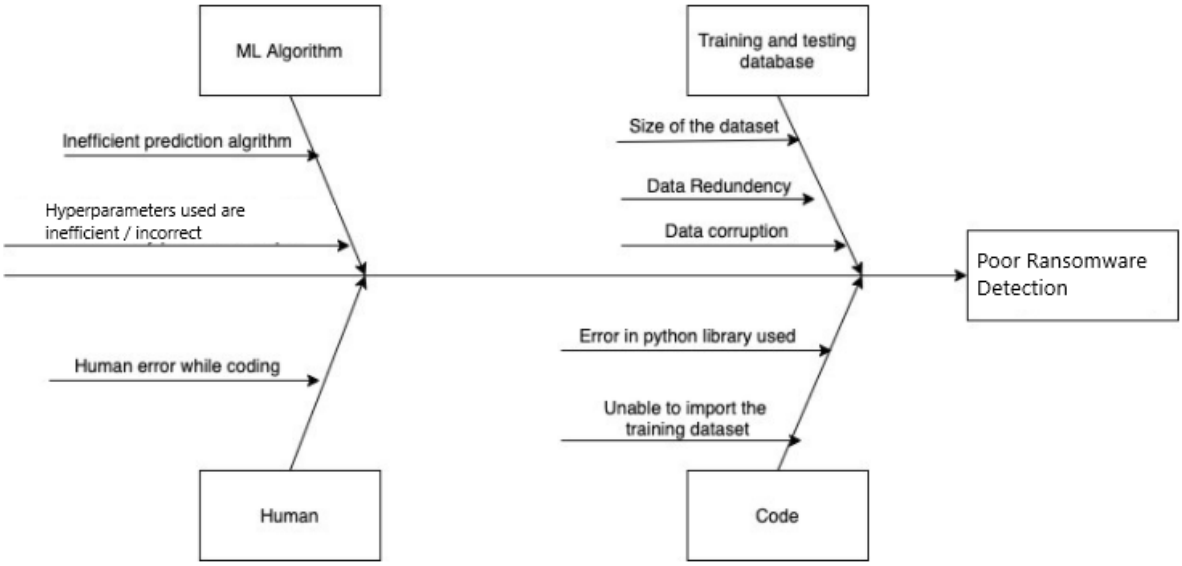

FIG 5.6 : FISHBONE DIAGRAM

7. MC CALL'S QUALITY FACTORS

1. Product Operation Factors

    - Correctness- The project aims at predicting the ransomwares with a high accuracy level.
    - Reliability- The degree to which the result of the prediction of ransomware is very reliable
    - Efficiency- The results of the prediction normally is having an efficiency of more than 88%
    - Integrity- It is a virtue which deals with proper utilization of software at every level and its association with the components of the system.

- Usability- The project is built in such a way that it is easily accessible and can be utilized by people of different strata of education and socio-economic status.

2. Product Revision Factors

    - Maintainability- The system aims to be up-to-date with time-to-time updates in software ,database and interface according to the needs and usage.
    - Flexibility- The project aims for easy yet accurate results for various ransomwares.
    - Testability- The system is easily testable by passing varying types of data to detect accuracy and mistakes.

3. Product Transition Factors

    - Portability- The system is very portable.
    - Reusability- The project holds a base strong enough for further improvement during its use and also acts as a starting milestone for other projects making use of it's features.
    - Interoperability- This project focuses on creating interfaces with other software systems or with other equipment firmware.

4. EXPERIMENT RESULTS & ANALYSIS

1. Results

The following results were observed:

- **KNN** achieved an accuracy of 95.7% but as it is a lazy learning algorithm its execution cost will tend to be greater on bigger datasets.

```
KNN
Accuracy =  0.951048951048951
Confusion Matrix =
[[ 72   0]
 [ 14 200]]
              precision    recall  f1-score   support

           0       0.84      1.00      0.91        72
           1       1.00      0.93      0.97       214

    accuracy                           0.95       286
   macro avg       0.92      0.97      0.94       286
weighted avg       0.96      0.95      0.95       286
```

FIG. 6.1.1 : KNN Results

- **Decision Tree** achieved an accuracy of 90.9%

```
Decision Tree
Accuracy =  0.9090909090909091
Confusion Matrix =
[[ 72   0]
 [ 26 188]]
              precision    recall  f1-score   support

           0       0.73      1.00      0.85        72
           1       1.00      0.88      0.94       214

    accuracy                           0.91       286
   macro avg       0.87      0.94      0.89       286
weighted avg       0.93      0.91      0.91       286
```

FIG. 6.1.2 : Decision Tree Results

- **Naive Bayes** achieved an accuracy of 90.2%

```
Naive Bayes
Accuracy =  0.9020979020979021
Confusion Matrix =
[[ 76   1]
 [ 27 182]]
              precision    recall  f1-score   support

           0       0.74      0.99      0.84        77
           1       0.99      0.87      0.93       209

    accuracy                           0.90       286
   macro avg       0.87      0.93      0.89       286
weighted avg       0.93      0.90      0.91       286
```

FIG. 6.1.3 : Naive Bayes Results

- **Random Forest** was able to achieve the highest accuracy of 97.2%. This was tuned with a higher number of trees to improve the accuracy and give more stable results, hence, impacted the performance slightly taking a longer duration (1.28s)

```
Random Forest
Accuracy =  0.972027972027972
Confusion Matrix =
[[ 74   3]
 [  5 204]]
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        77
           1       0.99      0.98      0.98       209

    accuracy                           0.97       286
   macro avg       0.96      0.97      0.96       286
weighted avg       0.97      0.97      0.97       286
```

FIG. 6.1.4 : Random Forest Results

2. Result Analysis

```
KNN: 0.965095 (0.012885)
Decision Tree: 0.905592 (0.008607)
Naive Bayes: 0.881250 (0.026862)
Random Forest: 0.979094 (0.017011)
```

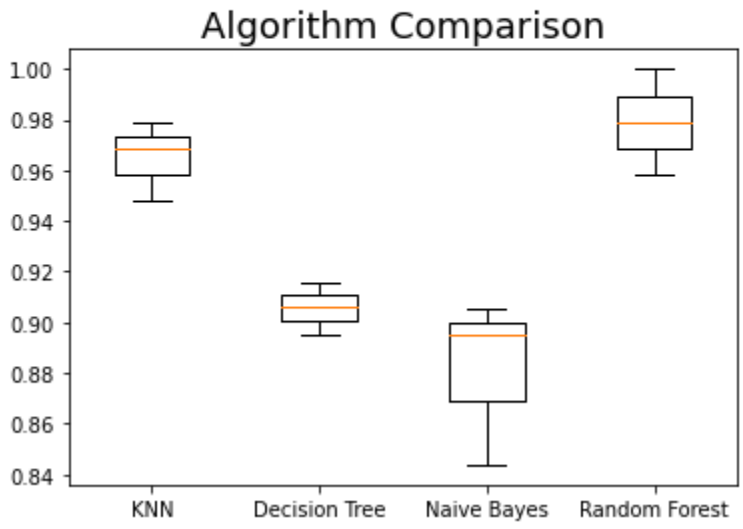

FIG 6.2.1 : ALGORITHM DATA COMPARISON

As we can see from the above results, the best performing model is KNN with an accuracy of 0.96% but has a high real time execution cost. Random Forest is able to perform the highest with an accuracy of 97.9% but this is at the cost of longer processing times. Compared to decision tree and naive bayes which are already tuned to best hyperparameters giving accuracies 0.90% and 0.88% respectively.

3. Conclusion and Future Work

Over the years, researchers have helped develop various strategies to aid in detecting and preventing malware attacks. Signature based detection is one of the most popular strategies used, but is challenged by newer and advanced malwares being churned out by hackers. Hence, an efficient and accurate dynamic based machine learning approach is required.

As we can conclude, from the results shown in the previous section:

- Random Forest is able to perform with higher accuracy compared to the other three models (Naive Bayes, KNN, Decision Tree). This model is able to distinguish between benign and infected files efficiently using the traces provided during the training and

validation process. The Random Forest model has an expensive cost of estimation trees, which is slightly heavier on the system, but provides a stronger and stable accuracy.

- Another good alternative to the random forest model is the KNN model, as it has also performed exceptionally well. Though, this KNN model is more suitable with smaller datasets due to high real time execution cost but allows for on the fly addition of data without affecting the accuracy of the system.

The future scope of this research is to implement the system in real time and test it thoroughly. This system can be further improved on, if paired with a signature based static method and will help detect malicious files earlier if code was not modified to bypass the signature based approach, but further research is required on that.

VII. PLAGIARISM REPORT

ORIGINALITY REPORT

1%	1%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.slideshare.net Internet Source	1%
2	Submitted to University of Hertfordshire Student Paper	<1%
3	Alan Ramirez-Noriega, Yobani Martinez-Ramirez, Jesus Chavez Lizarraga, Kevin Vazquez Niebla, Jesus Soto. "A software tool to generate a Model-View-Controller architecture based on the Entity-Relationship Model", 2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT), 2020 Publication	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		

VIII. CONFERENCE CERTIFICATE

Conference Name: International Conference on Recent Advancements in Interdisciplinary Research.

Paper Id: 15

Venue and Date: 29th and 30th May, Asian Institute of Technology Conference Center, Pathumthani, Thailand.

Title: Improving Ransomware Techniques using Machine Learning.

To be published in: Scopus/SCI/ESCI/Web of Science indexed international journal.

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

IshanUdaiChaudhari RA1711020010096 <iu7825@srmist.edu.in>

Registration Procedure for Paper ID : 15 in 9th Online Conference ICRAIR 2021

1 message

9th ICRAIR 2021 <9thicrair2021@easychair.org> Fri, Apr 30, 2021 at 12:41 PM
To: Ishan Chaudhari <iu7825@srmist.edu.in>

Dear Ishan Chaudhari,

Paper-ID:15

Title: Improving Ransomware Detection Techniques using Machine Learning

Authors: Ishan Chaudhari, Varshini Venkatesh, Dr. Usha G.

Greetings of the day!!!!

Congratulations, your manuscript titled –Improving Ransomware Detection Techniques using Machine Learning has been ACCEPTED for publication in 9th International Online Conference on Recent Advancements in Interdisciplinary Research (ICRAIR 2021) to be held on 29-30 May 2021 at Asian Institute of Technology Conference Center, Pathumthani, Thailand.
All accepted and registered papers will be published in SCOPUS/SCI/ESCI/Web of Science indexed International Journal*
Kindly complete your registration process using the following Link:
http://icrair.com/registration

Last date for early bird registration is 10th May 2021.

After registration, kindly send the following documents in one single folder to icrair09@gmail.com with caption:

1. Camera Ready Paper (PaperID _camerar.DOC or PaperID _camerar.DOCX)
Download Paper format: http://icrair.com/download/paper-format
2. Copyright Form duly signed by all the authors(PaperID _copyr.PDF)
Download Copyright form: http://icrair.com/download/copyright-form
3. e-receipt of Fund Transfer.
Mode of Payment:
you can pay through Western Union in Name of Rajkumari, India
Registration fees+Transaction Charges
OR
Direct link mentioned on http://icrair.com/registration through payumoney/paypal payment gateway.
Registration fees+Transaction Charges

*Note:
1. Revision of paper may be required at later stage.
2. Selection of journal for publication of your paper will be done by board members of ICRAIR.
3. Plagiarism of Paper must be less than 15%
4. Additional Information
For any further queries regarding registration, please contact
Dr. Aman Gupta, E-mail - icrair09@gmail.com (Mb. +91-9870398104).

Thank you very much & we look forward to see you at 9th ICRAIR 2021.

--
Best wishes,
Organizing Chair
9th ICRAIR 2021
www.icrair.com

IX. REFERENCE

[1] Brijesh Jethva, Issa Traoré, Asem Ghaleb, Karim Ganame, Sherif Ahmed, **"Multilayer Ransomware Detection using Grouped Registry Key Operations, File Entropy and File Signature Monitoring"** Journal of Computer Security, **Volume 28, Number 3, 2020, pages 337-373.**

[2] Martina Jose Mary.M1 , Usharani.S², Thirugnanam.P³, **"Detection and Deterrence of Ransomware using Machine Learning Techniques: A survey"** , Journal of Information and Computational Science, **Volume 10 Issue 1 – 2020**.

[3] Ban Mohammed Khammas, **"Ransomware Detection using Random Forest Technique"**, ICT Express, **volume 6, issue 4, December 2020, Pages- 325-331.**

[4] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, Mikel Izal, **"A Survey on Detection Techniques for Cryptographic Ransomware"** , IEEE Access,Volume 7, pages-144925 – 144944, 07 October 2019.

[5] Cuneyt Gurcan, Yitao Li, Yulia R. Gel, Murat Kantarcioglu, **" BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain "**, arXiv:1906.07852v1 **[cs.CR]** , 19 june 2019.

[6] Li Chen, Chih-Yuan Yang, Anindya Paul, Ravi Sahita, **" Towards Resilient Machine Learning for Ransomware Detection"** , arXiv:1812.09400v2 **[cs.LG]** ,2018.

[7] Edgar P. Torres P. and Sang Guun Yoo, **" Detecting and Neutralizing Encrypting Ransomware Attacks by Using Machine-Learning Techniques: A Literature Review "**, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp. 7902-7911.

[8] Samah Alsoghyer, Iman Almomani, **"Ransomware Detection for Android Applications"**, *Electronics* **2019**, *8*(8), 868; **https://doi.org/10.3390/electronics8080868**

[9] S. H. Kok , Azween Abdullah , N. Z. JhanJhi, Mahadevan Supramaniam , **"Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm"**, *Computers* **2019**, *8*(4), 79; **https://doi.org/10.3390/computers8040079**

[10] A Deep Neural Network Strategy to Distinguish and Avoid Cyber Attack, Siddhant Agarwal, Abhay Tyagi ,G. Usha, Artificial Intelligence and Evolutionary Computations in

Engineering Systems pp 673-681, Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 1056).

[11] J. Jeyasudha and G. Usha, **"Detection of Spammers in the Reconnaissance Phase by machine learning techniques,"** 2018 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2018, pp. 216-220, doi: 10.1109/ICICT43934.2018.9034457.

[12] E. S. Domi and G. Usha, **"TSDLA: Algorithm for Location Privacy in Clustered LB-MCS network,"** *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2020, pp. 68-74, doi: 10.1109/ICSSIT48917.2020.9214260.