PROJECT REPORT ON

A Machine Learning Framework for DomainGeneration Algorithm (DGA)-Based Malware Detection

SUBMITTED BY

Harsh Dobariya

Akshay Kalapgar

Mohit Kamble

Siddesh Parab

GUIDED BY

Prof. Abhay E. Patil

MANJARA CHARITABLE TRUST

RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

(Permanently Affiliated to University of Mumbai)

Juhu Versova Link Road, Andheri (West), Mumbai-53

DEPARTMENT OF INFORMATION TECHNOLOGY

UNIVERSITY OF MUMBAI

2021

MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI
(Permanently Affiliated to University of Mumbai)
Juhu Versova Link Road, Andheri (West), Mumbai-53

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

Date: _____

This is to certify that, the project work embodied in this report entitled, *"**A Machine Learning Framework for Domain Generation Algorithm (DGA)-Based Malware Detection**"* submitted by "***Harsh Dobariya*** bearing Roll No. 814", "***Akshay Kalapgar*** bearing Roll No. 831", "***Mohit Kamble*** bearing Roll No. 832", "***Siddesh Parab*** bearing Roll No. 850"for the award of ***Bachelor of Engineering*(B.E.)** degree in the subject of ***Information Technology***, is a work carried out by them under my guidance and supervision within the institute. The work described in this project report is carried out by the concerned students and has not been submitted for the award of anyother degree of the University of Mumbai.

Further, it is certifying that the students were regular during the academic year 2020-20 and have worked under the guidance of concerned faculty until the submission of this project work at ***Rajiv Gandhi Institute of Technology, Mumbai.***

Prof. Abhay E. Patil	Prof. Swapnil Gharat
Project Guide	**Project Coordinator**
Dr. Sunil B. Wankhade	Dr. Sanjay U. Bokade
Head of Department	**Principal**

CERTIFICATE OF APPROVAL

This project report entitled

A Machine Learning Framework for Domain Generation Algorithm(DGA)-Based Malware Detection

Submitted by:

HARSH DOBARIYA 814

AKSHAY KALAPGAR 831

MOHIT KAMBLE 832

SIDDESH PARAB 850

In partial fulfillment of the requirements of the degree of **Bachelor of Engineering** in **Information Technology** is approved.

Internal Examiner

_____

SEAL OF INSTITUTE

External Examiner

_____

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

ROLL NO.	NAME	SIGNATURE
814	HARSH DOBARIYA	
831	AKSHAY KALAPGAR	
832	MOHIT KAMBLE	
850	SIDDESH PARAB	

Date:

Place:

Acknowledgement

With all reverence, we take the opportunity to express our deep sense of gratitude and wholehearted indebtedness to our respected guide, **Prof. Abhay E. Patil**, Department of Information Technology, Rajiv Gandhi Institute of Technology, Mumbai. From the day of conception of this project his active involvement and motivating guidance on day-to-day basis has made it possible for us to complete this challenging work in time.

We would like to express a deep sense of gratitude to our respected **Head of the Department**, **Dr. Sunil B. Wankhade** who went all the way out to help us in all genuine cases during the course of doing this project. We wish to express our sincere thanks to **Dr. Sanjay U. Bokade**, **Principal**, Rajiv Gandhi Institute of Technology, Mumbai and would to like to acknowledge specifically for giving guidance, encouragement and inspiration throughout the academics.

We would like to thank all the staff of Information Technology Department who continuously supported and motivated during our work. Also, we would like to thank our colleagues for their continuous support and motivation during the project work. Finally, we would like to express our gratitude to our family for their eternal belief in us. We would not be where we are today without their support and encouragement.

HARSH DOBARIYA

AKSHAY KALAPGAR

MOHIT KAMBLE

SIDDESH PARAB

Date:

Place:

ABSTRACT

Attackers usually use a Command and Control (C2) server to manipulate the communication. In order to perform an attack, threat actors often employ a Domain Generation Algorithm (DGA), which can allow malware to communicate with C2 by generating a variety of network locations. Traditional malware control methods, such as blacklisting, are insufficient to handle DGA threats. In this paper, we propose a machine learning framework for identifying and detecting DGA domains to alleviate the threat. We collect real-time threat data from the real-life traffic over a one- year period. We also propose a deep learning model to classify a large number of DGA domains. The proposed machine learning framework consists of a two level model and a prediction model. In the two-level model, we first classify the DGA domains apart from normal domains and then use the clustering method to identify the algorithms that generate those DGA domains. To differentiate DGA domain names from normal domain names, researchers have discovered that DGA-generated domain names contain significant features. Therefore, many studies aim to target blocking those DGA domain names as a defense approach. The DGA that generates the domain fluxing botnet needs to be known so that we can take countermeasures. Several studies have looked at understanding and reverse engineering the inner workings of botnets. The study focused on domain fluxing malware and relied on the binary extraction for DGA. Their approach is only effective for certain types of malware. To be precise, we achieve an accuracy of 95.89% for the classification in the framework and 97.79% in the DNN model, 92.45% for the second-level clustering, and 95.21% for the prediction in the framework.

TABLE OF CONTENTS

ABSTRACT vi

TABLE OF CONTENTS vii

LIST OF FIGURES ix

LIST OF TABLES x

Chapter 1 Introduction 1

Introduction to Project 1

Background of the Study 2

Statement of the Problem 2

Objectives of the Project 3

Purpose of Study 3

Scope of the work… 4

Limitation of the Project 6

Chapter 2 Literature Review 7

Technical Paper Review 7

Existing System 10

Chapter 3 Methodology 11

Proposed System 11

Principle of working 15

Filtering packet data and feature extraction work 16

Machine Learning Classification 16

Testing 18

System Requirements 21

Software Requirements 21

Hardware Requirements 21

Chapter 4 Working of the system 22

Beginning with basic system application 22

Training the model 23

Plotting the confusion matrix 24

First test prediction for a single domain 24

Result for first test prediction for a single domain 25

Second test prediction for a single domain 26

Result for second test prediction for a single domain 26

Prediction of a list of domains 27

Result for prediction a list of domains 27

Chapter 5 Conclusion and Future Work 29

Conclusion 29

Future Work 29

REFERNCES 30

LIST OF FIGURES

Figure 3.1 Diagrammatical Representation of working of the system.....11

Figure 3.2 Flowchart of the system 12

Figure 3.3 Class diagram of the system 13

Figure 4.1 Basic System Application.....22

Figure 4.2 Training the model. 23

Figure 4.3 Confusion Matrix 24

Figure 4.4 First Test Prediction for a single domain 25

Figure 4.5 First test result..... 25

Figure 4.6 Second test for a single domain prediction.....26

Figure 4.7 Second test result 26

Figure 4.8 Prediction for a list of domains.....27

Figure 4.9 File location of result for a list of domains.....28

Figure 4.10 CSV file of result for a list of domains.....28

LIST OF TABLES

Table 2.1 Technical Paper Review 7

CHAPTER 1
INTRODUCTION

Introduction to Project

Malware attackers attempt to infiltrate layers of protection and defensive solutions, resulting in threats on a computer network and its assets. Anti-malware software have been widely used in enterprises for a long time since they can provide some level of security on computer networks and systems to detect and mitigate malware attacks. However, many anti-malware solutions typically utilize static string matching approaches, hashing schemes, or network communication white listing. These solutions are too simple to resolve sophisticate malware attacks, which can hide communication channels to bypass most detection schemes by purposely integrating evasive techniques. The issue has posed a serious threat to the security of an enterprise and it is also a grand challenge that needs to be addressed.

In this project, we first propose a machine learning framework to classify and detect DGA malware and develop a DNN model to classify the large datasets of DGA domains that we gradually collected. We then experimentally evaluate the proposed framework through a comparison of various machine learning approaches and a deep learning model. The general goal of our machine learning framework is to determine which algorithm is employed so that our proposed framework can prevent future communications from the C2.

The comparison results provide us a useful guideline for our future study in DGA detection and prediction. In our future research, we will also apply deep learning in clustering and predictions that are out of the scope of this paper.

Background of the study

Some of the sophisticate malware attackers use either a static or dynamic method to communicate with a centralized server to service a Command and Control (C2). In a static method, everything is fixed. For example, the malware has both a fixed IP address and a fixed domain name permanently (i.e., its domain name will not change throughout its lifespan). Thus, as long this malware has been identified as a threat, a simple rule can be applied to resolve this malware threat issue.

In a dynamic method, Domain Generation Algorithm (DGA) has been commonly used to communicate back to a variety of servers. The DGA is a sequencing algorithm that is used to periodically generate a large number of domain names, which are often used by malware to evade domain-based firewall controls.

The generated domain names give malicious actors the opportunity to hide their C2 servers so that it is hard for the enterprise to identify the DGA. The domains generated by DGAs are short-lived registered domains and they are easier for human to identify but harder for machines to detect automatically.

Statement of the Problem

Multiple conditions for a DGA to function in a network environment where filtering results in a firewall that protects the communication and an empty cell in an Internet domain that result in an NXDOMAIN error.

Firewall blacklisting constantly expands as the multiple sources of inputs expand filtering rules. However, sequences in a DGA may not be known to these inputs promptly. Moreover, for the malware that communicates with an appropriate domain correctly, a threat actor must register each respective domain name in the sequence to maintain the C2 or risk the loss of a node in the distribution.

Our research problem is to accurately identify and cluster domains that originate from known DGA-based techniques where we target to develop a security approach that autonomously mitigates network communications to unknown threats in a sequence

Objectives of the Project

The main objective of our project is to create a malware detection system, which can test analyzeand verify each and every domain using a training data set, and provide an outcome whether the domain contains malware or not. This system will help the networking institutions take effective and timely decisions, replacing the existing time taking, long procedures and paper work required to check the credibility of the domains. The system aims to achieve:

- In DBSCAN algorithm, we use the features described above to calculate the domain distance and to group the domains that are generated by the same DGA together according to their domain feature difference.
- Distinguish the model from training and prediction stages.
- The nodes in each layer are fully connected to the nodes in the next will not miss any local minima, but it will take a long time to converge.

Purpose of study

The purpose of our work is to present a data analysis model for effective classification among domains, and enhanced decision making in malware detection to the applicants. For users, malware risk is a major challenge, which directly or indirectly affects the reliability of the network.

The model allows storage of the huge volume of domain data, which is then cleaned and features are extracted and reduced.

Scope of the work

The networking industry is among many industries which have massive and useful data about their customers but very few are utilizing this set of information to enhance the user experience and using the data information to prevent fraud. The challenge is not about dealing with trillions of bytes of data, it is about getting started with a quantitative approach so that you can drive value from your data, whatever size that data is. They are very well aware of the fact that if the data can be used effectively they can fulfill the needs of user accurately.

Major scope of malware detection in networking industry in the present and near future includes:

- **User Segmentation**

  User segmentation is classifying the customer on the basis of the age, gender, behavior, habits etc. The networking industry has agreed that customer retention is a keyto company's success and is becoming more user-centric.

  The analysis helps the networking services to analyze the spending pattern of an individual user which help them to offer services time to time to their users. The analysis also helps in identifying a valuable user, one who spent the most money. And through this data analysis, they can provide best to the user. This ultimately will lead to increased user satisfaction. Additionally, it will also help the networking industry understand the spending patterns of the users, domain usage, and consequently cross-selling of various domains.

- **Malware Detection**

  This is one of the biggest problems that every networking industry has been facing. With the increase in an online transaction, the incidents of fraud have increased too. To avoid such fraud the networking industry is using the malware detecting technology which helps

them to understand the domain history and using pattern of users and increase security on every unusual transaction. This will help them to mitigate any fraudulent activities before it grows bigger.

- **Offering Personalized Services**

  Offering Personalized Services to a user is nothing but the next level of marketing where they offer product and services to as per users' interest and requirement. Yes, it is possible with the help of data analysis. Networking industry collects data from e-commerce website and malware detecting technology analyze the buying habit, interest and requirements of individual user by doing data analysis. This data analysis helps to offer services and products to the user time to time as per their interest and requirements which help them to retain the present user and attract the new one.

- **Risk Management**

  Risk Management is an important factor in every industry and risk in the networking industry can come in any form like an unrecoverable fees after malware detection, and fraudulent activities. It cannot be stopped completely but the early detection of risk can be helpful in preventing huge losses. With the help of data analysis, they can perform the risk management analysis and can minimize the user's risk.

- **Addressing Compliance Requirements**

  Networking services are required to do regular compliance, audit and maintain certain regulations for their data, privacy and security measures. They now have access to billions of user's needs. They can use data to cater to serve the user more effectively. Cloud-based analytics packages can sync in real time with your data systems, creating actionable insight dynamically.

Limitation of the Project

The limitation of the project is based on the training of dataset and accuracy achieved credit analysis.

- For training of large dataset, high computation system is required which can handle large scale processing.
- Domain generation algorithms are not intended to replace the data entities; they just try to show the malware-detecting process.
- It not possible to determine in advance the accuracy of the developed algorithms.
- Resistance to face new challenges related with privacy and deal with many users, technologies and data sources.

CHAPTER 2

LITERATURE REVIEW

Technical Paper Review

The amount of research carried out and successfully implemented in the field of Machine Learning is currently limited due to novelty of the technology. This section elaborates an extensive study of the available research in order to gather enough information to meet our objectives.

Table 2.1 Technical Paper Review

Title	Authors	Advantages	Disadvantages	Result
[1] A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection.	Yi Li, Kaiqi Xiong, Tommy Chin, Chengbin Hu. Institute of Electrical and Electronics Engineers, 2019.	In the second-level clustering we apply the DBSCAN algorithm. Only the DGA domains obtained from the first-level classification will be used for clustering.	Research problem is to accurately identify and cluster domains that originate from known DGA-based techniques where we target to develop a security approach that autonomously mitigates network communications to unknown threats in a Sequence.	Domain Generation Algorithm (DGA) is used.

[2] Learning and Classification of Malware Behavior.	Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, Pavel Laskov. Kluwer Academic Publishers, Dordrecht (2002).	Results show that 70% of malware instances not identified by an anti-virus software can be correctly classified by our approach.	Proposed machine learning framework aims to solve the problem of detecting DGA sequences using machine learning techniques derived from observations in a network.	Normalize Compression Distance. Benign Executable
[3] An SDN based framework for guaranteeing security and performance in information-centric cloud networks.	U. Ghosh, P. Chatterjee, D. Tosh, S. Shetty, K. Xiong, and C. Kamhou. 11th IEEE International Conference on Cloud Computing (IEEE Cloud), 2017.	The framework has the ability to dynamically compute the routing path to guarantee security and performance of the network.	Queries not matching the knowledge are stored in a backlog of the software.	SDN-based framework and Information centric services

[4] A two-hashing table multiple string pattern matching algorithm.	C. Khancome, V. Boonjing, and P. Chanvarasuth. Tenth International Con- ference on Information Technology: New Generations (ITNG). IEEE, 2013.	The attempting times were less than of the traditional algorithms especially in the case of a very long minimum pattern length.	The lengthy processing time when directly extended to the multiple string patterns matching.	Executes multiple string pattern matching algorithms.

Existing System

Threat models: Multiple conditions for a DGA to function in a network environment where filtering results in a firewall that protects the communication and an empty cell in an Internet domain those results in NXDOMAIN error.

Each HMM date record represents a series of domain observations. First sequences of domain name are processed by a feature extractor and each of these feature vectors is used as a training record.

Then, similar sequences are clustered as a group of DGA domain names with certain outcomes. After the training process, if a sequence does not have an HMM sequence representation (or it is not presented in the training data but the test data), the HMM model then generates the future predicted results. Otherwise, we will use an existing HMM sequence presentation.

Disadvantages of Existing System:

1. Firewall protects the communication and an empty cell in an internet domain that results inno domain error.
2. Queries not matching the knowledge are stored in a backlog of the software.

CHAPTER 3

METHODOLOGY

3.1 Proposed System

In our proposed system, Domains extracted from DGAs. Machine learning framework that encompasses multiple feature extraction techniques and the models to classify the DGA domains from normal domains, cluster the DGA domains, and predict a DGA domain.

A deep learning model to handle large datasets multiple on- line sources from simple Google searching provide example codes for a DGA construction.

Online threat intelligence feeds give an approach to examining current and live threats in real-world environment.

Using real-time active malicious domains derived from DGAs on the public Internet measures the accuracy of the proposed approach. The structure of the data is presented in a CSV format of domain names, originating malware, and DGA membership with the daily file size of approximate 110MB.

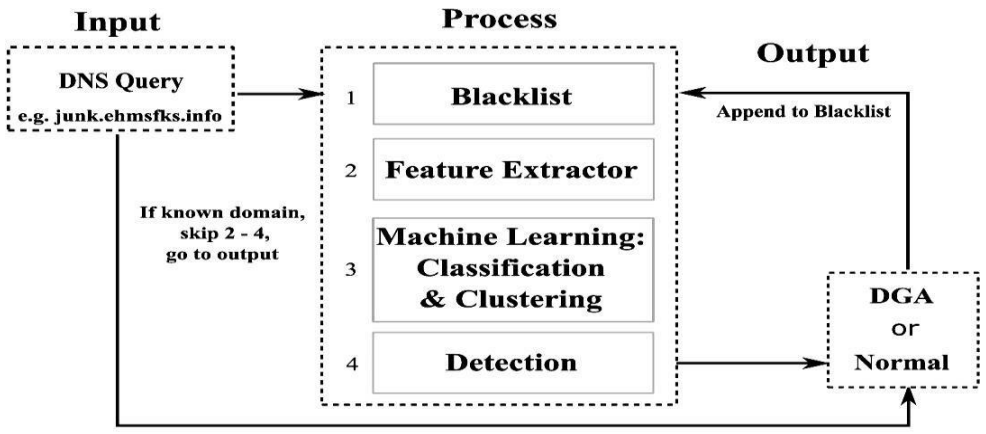

Fig. 3.1: Diagrammatical Representation of working of the system

Flowchart of the System

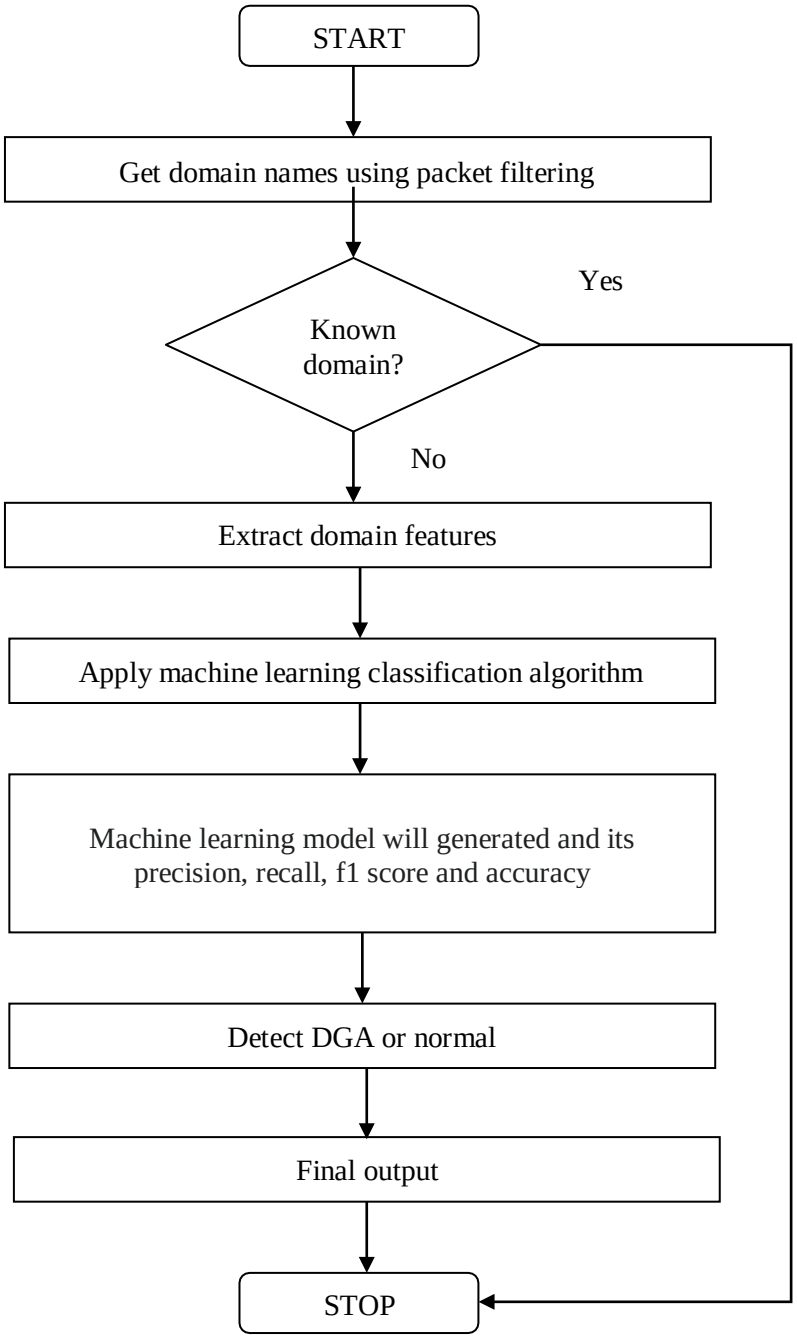

Fig. 3.2 - Flowchart of the System

Class diagram of the System

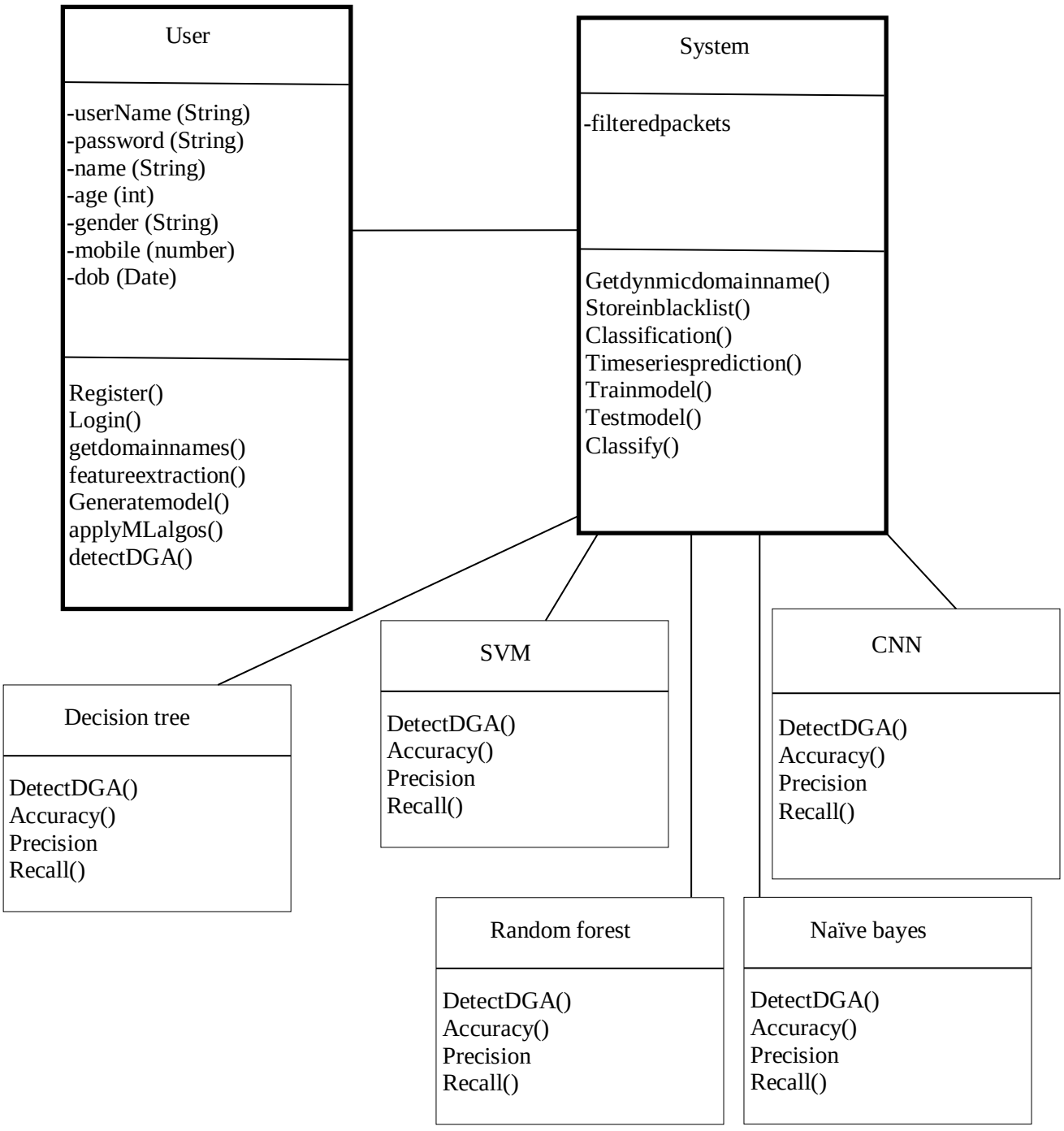

Fig. 3.3 – Class diagram of the System

In the project, we have assumed that DGA domains have groups of very significant characters from normal domains. By grouping domains according to their features, the authors applied a machine learning classifier to distinguish DGA domains from normal domains easily.

We propose a machine learning framework that consists of three important steps, as shownin Figure below. We first have the DNS queries with the payload as the input. In order to classify DGA domain names.

Advantages of Proposed System:

1. Domain Generation Algorithm (DGA), which allows malware to generate numerous domainnames until it finds its corresponding C&C server.
2. It is highly resilient to detection systems and reverse engineering, while allowing the C&Cserver to have several redundant domain names.

3.2 Principle of working

We first have the DNS queries with the payload as the input. Then, the DNS queries will be passed to our process step, which consists of 4 important components:

1. We first use a domain-request packet filter to get domain names and then store them in a dynamic blacklist. If the input is a known domain, we will skip (2) - (4) and directly go to the output; otherwise, we will proceed to the next component.
2. Then, a feature extractor is used to extract domain features.
3. Next, we apply the first-level classification to distinguish DGA domains from non-DGA domains and the second-level clustering to group similar DGA domains.
4. Finally, we use a time-series model to predict the features of a domain. After the domain name goes through the process step, we will append this domain to the dynamic blacklist. The rest of this section discusses the four components of the process step in details.

3.3 Filtering packet data:-

To filter packet data we are using pyshark which captures network packets.We will store this packet information in pcap format
By reading packet we will filter the data and obtain domain name.Packet flow also obtained from this.
If domain name extracted in this found in blacklist we will stop further steps.

3.4 Feature extraction work:-

With the python coding we will calculate the following feature

- Length- length of domain name.
- Meaningful Word Ratio,:- dictionary will be maintained of meaningful word and output willbe taken by dividing with length of domain name
- Percentage of Numerical Characters,:- numeric character involved in domain name system.
- Pronounce ability Score—frequency of text in domain calculated.
- Percentage of the Length of the Longest Meaningful String (LMS):- dividing the meaningfulword with the length of domain.
- Levenshtein Edit Distance:- It measures the minimum number of single-character edits between a current domain and its previous domain in a stream of DNS queries received by the server. The Levenshtein distance is calculated based on a domain and its predecessor

3.5 Machine learning classification:-

Following algorithms will be applied on feature obtained above.

- **Decision Tree:-** It calculates entropy and information gain and output generated but has problem of over fitting. We will generate module with the selected feature.

- **ANN:-** It's Artificial Neural Networks. Here we give input layer, hidden layer and output layer. Then with the feature we calculate the output.

- **Multiple Logistic regression:-** the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.
- **Naive Bayes:-** it calculates probability of certain class. Model will be generated using pickle and stored
- **Random Forest:-** Random forest avoids over fitting problem and model will be generated, stored into pickle.

All this machine learning model will generated and its

i. precision
ii. recall
iii. f1 score
iv. Accuracy will be calculated.

- **Clustering:-**

Dbscan used for outlier's detection

Outliers are specific entries in dataset that are different than other point and don't play vitalrole in classification.

In statistics, an **outlier** is an observation point that is distant from other observations.

In this domain name will be clustered based on:

i. Cryptolockereg. nxgbdtnvrfker.ru
ii. TOVAReg.:- gppwkpxyremp.net
iii. Dyreeg:- q2aa41a5b31294e5e6f28d1adcf48a54b.tk
iv. normalDomaineg:- easypdfcombine.com

3.6 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide anobjective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its function within an Acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusive) attempts to execute a programor application with the intent of finding software bugs (errors or other defects).

The job of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones. Software testing can provide objective, independent information about the quality of software and risk of its

failure to user and/or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how the testing is conducted. For example, in a phased process, most testing occurs after the system requirements have been defined and then implemented in testable programs. In contrasts, under an Agile approach, requirements, programming, and testing are often done concurrently.

LEVELS OF TESTING:

UNIT TESTING

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In proceduralprogramming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object- oriented programming, a unit is often an entire interface, such as a class,but could be an individual method, unit tests a short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against software design. Software components may be integrated in an iterative wayor all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

REGRESSION TESTING

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regressions, as degraded

or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly, stops working was intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include rerunning previous sets of test cases and checking whether previously fixed faults have re- emerged.

SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within thescope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

VALIDATION TESTING

Validation Testing ensures that the product actually meets the client's need. It can also be definesas to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

System Requirements

Software Requirements

- Spark
- Python 3

Hardware Requirements

- Minimum RAM required: 4GB (Suggested: 8GB)
- Minimum Free Disk Space: 25GB
- Minimum Processor i3 or above
- Operating System of 64bit

CHAPTER 4

WORKING OF THE SYSTEM

In this section, we explain the working of our proposed system illustrated with the help of screenshots shown below.

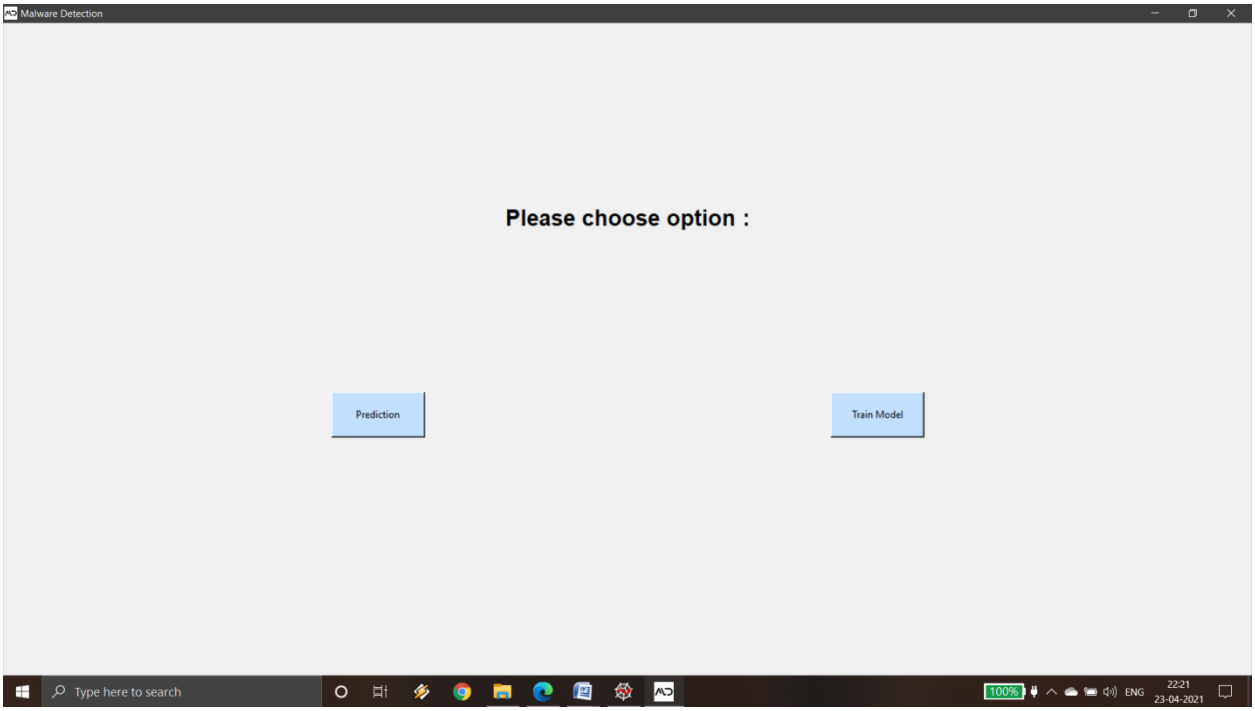

Figure 4.1: Basic System Application

Beginning with basic system application

It is a basic system application made with the help of Tkinter GUI.

Here there are two buttons where the user can choose to train the model and get the confusion matrix and classification report or to predict the domain(s) whether they contain malware or not.

Training the model

By clicking on the “Train Model” button the user is directed towards the next page of the application where the user can input the dataset file’s path and start the prediction.

Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behavior of biological systems composed of “neurons”. A neural network is a machine learning algorithm based on the model of a human neuron. The ANN algorithm after comparing with the other mentioned algorithms showed the best accuracy and so we used it for the UI.

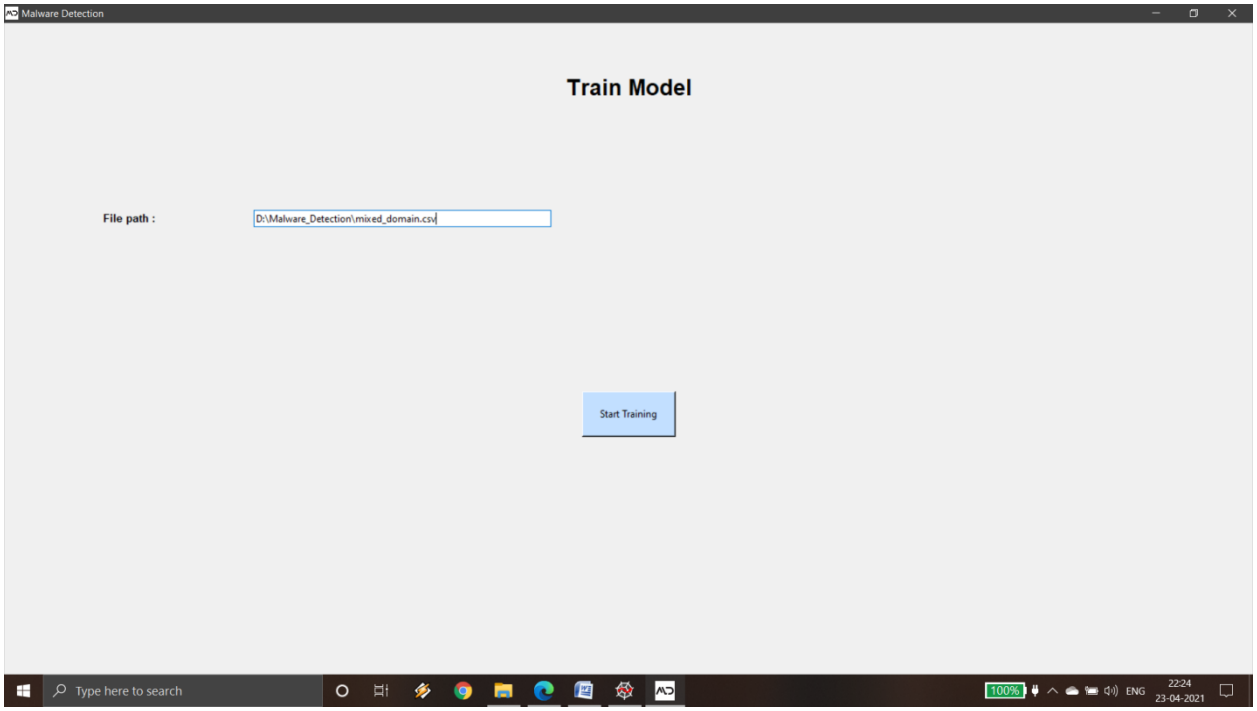

Figure 4.2: Training the model.

Plotting the confusion matrix

Spark is a lightning-fast cluster computing technology, designed for fast computations which include interactive queries and stream processing. The classification reports are present in the logs section of spark and the confusion matrix is obtained in the system application. Also the whole model gets store in a sub folder where the main program file is present.

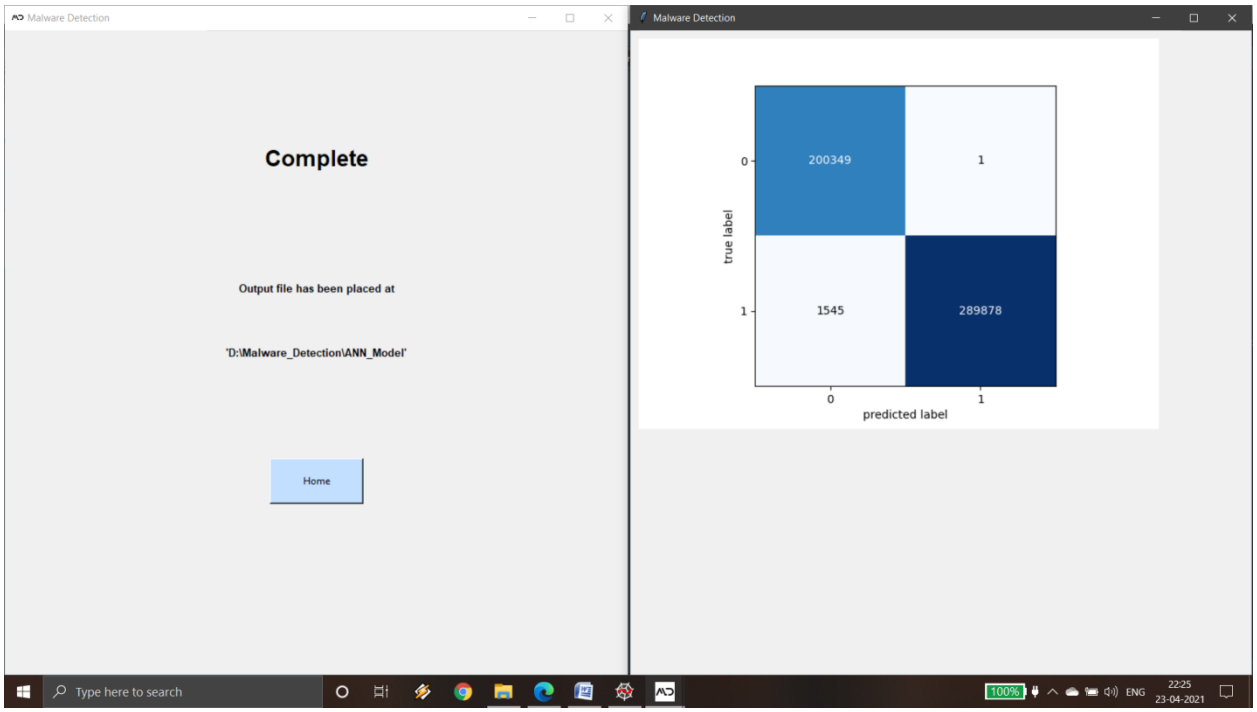

Figure 4.3: Confusion Matrix

First test prediction for a single domain

If the user wants to predict a single domain, the family of the domain (if any) should be entered in the textbox aside “DGA Family” label.

After that enter the domain in the textbox aside the “Domain” label and start the prediction.

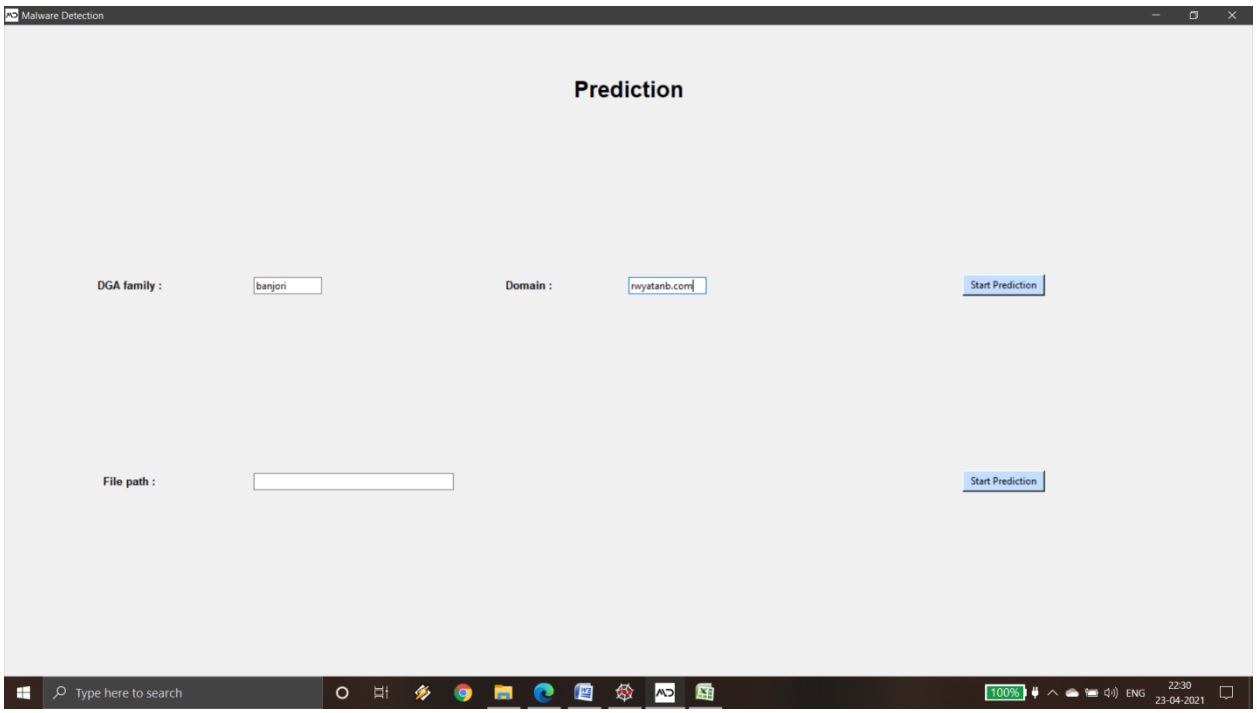

Figure 4.4: First Test Prediction for a single domain.

Result for first test single domain prediction.

Here the first test shows that the domain contains malware.

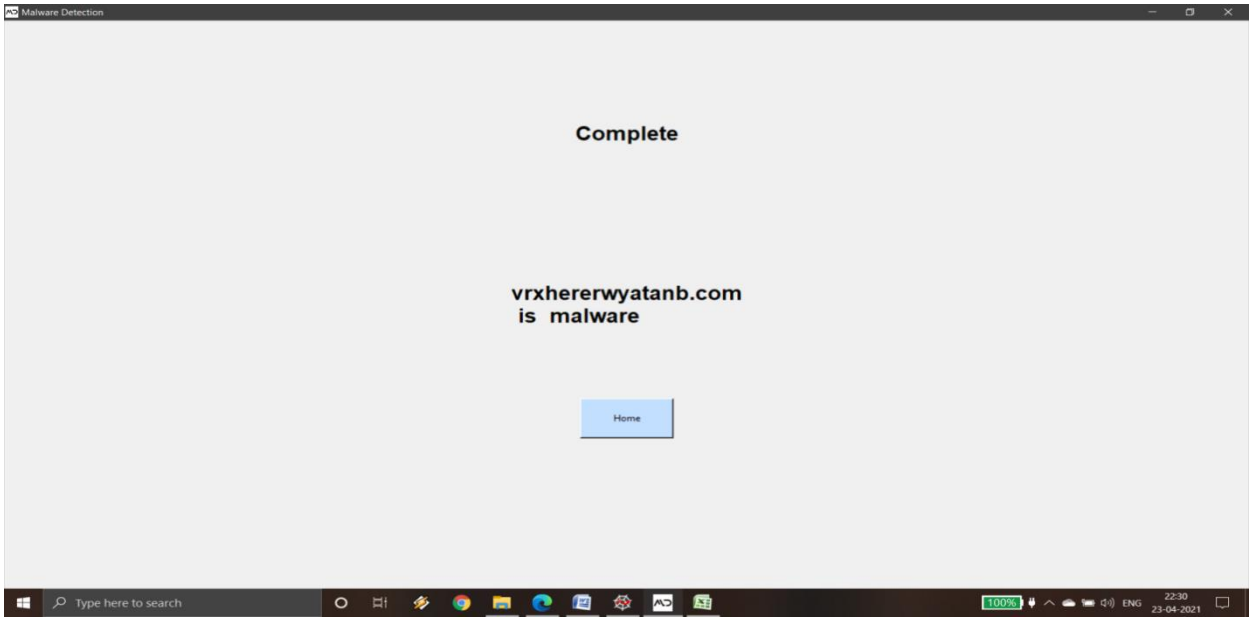

Figure 4.5: First test result

Second test for a single domain prediction

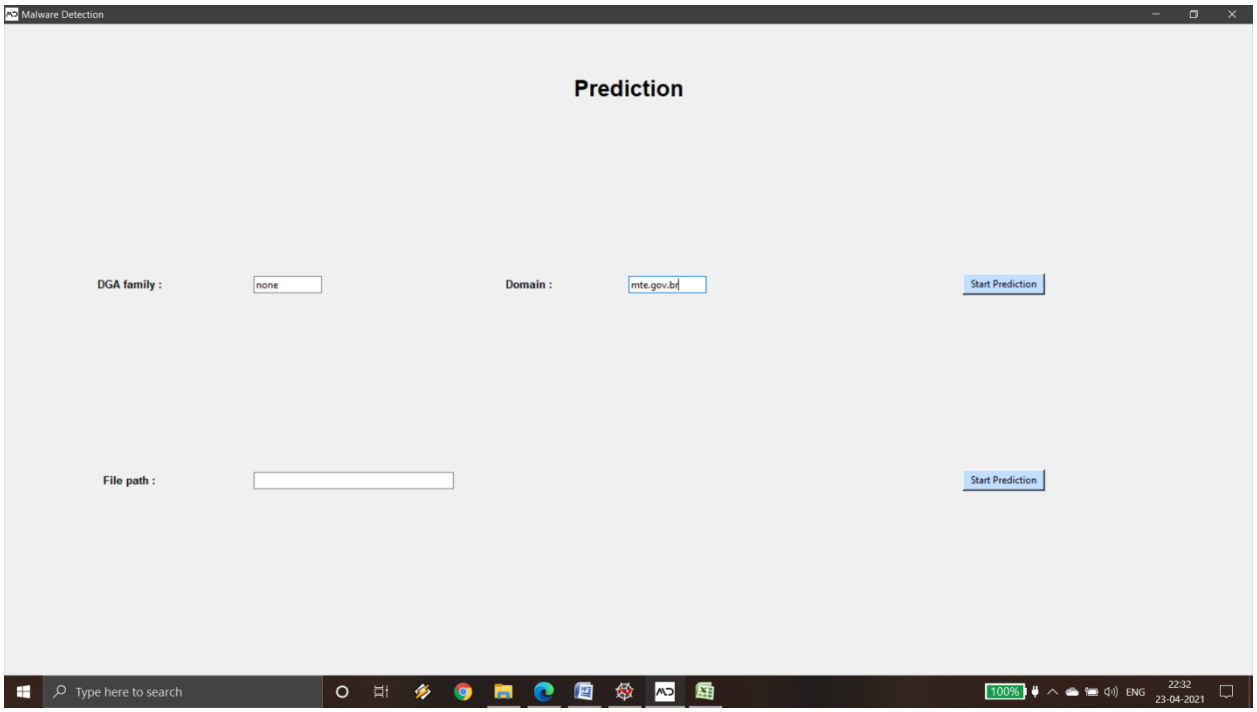

Figure 4.6: Second test for a single domain prediction.

Result for second test for a single domain prediction.

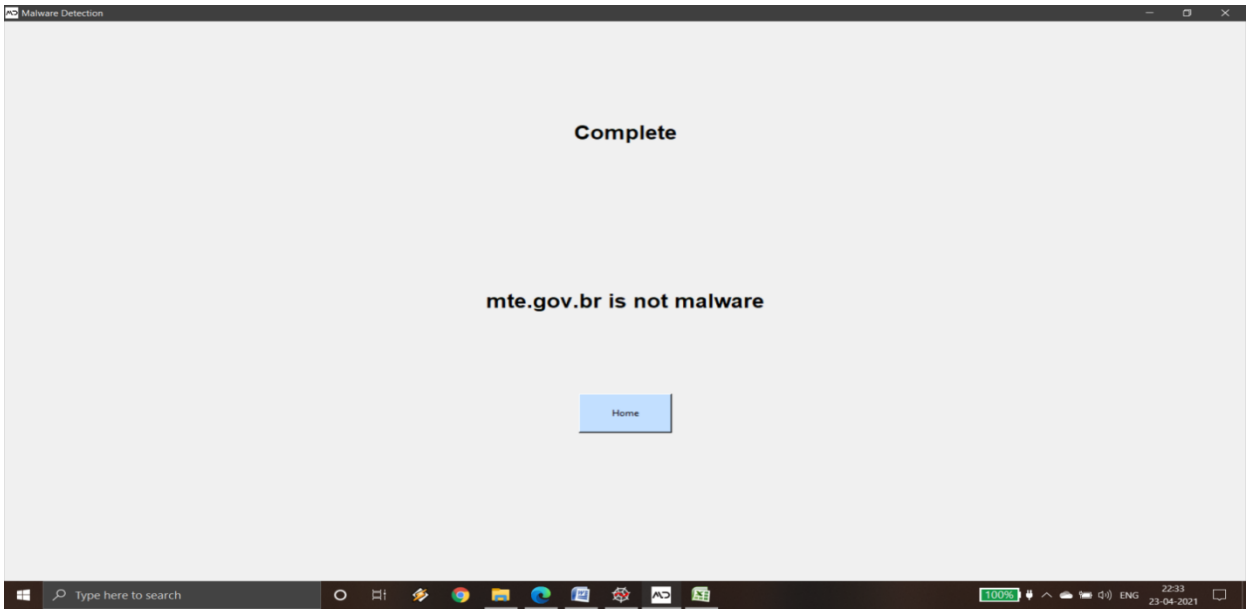

Figure 4.7: Second test result

Prediction for a list of domains

Here if the user wants to predict a list of domains, the user can input the file's path containing the list of domains in the textbox aside "File path" label and start the prediction by clicking the "Start Prediction" aside the textbox.

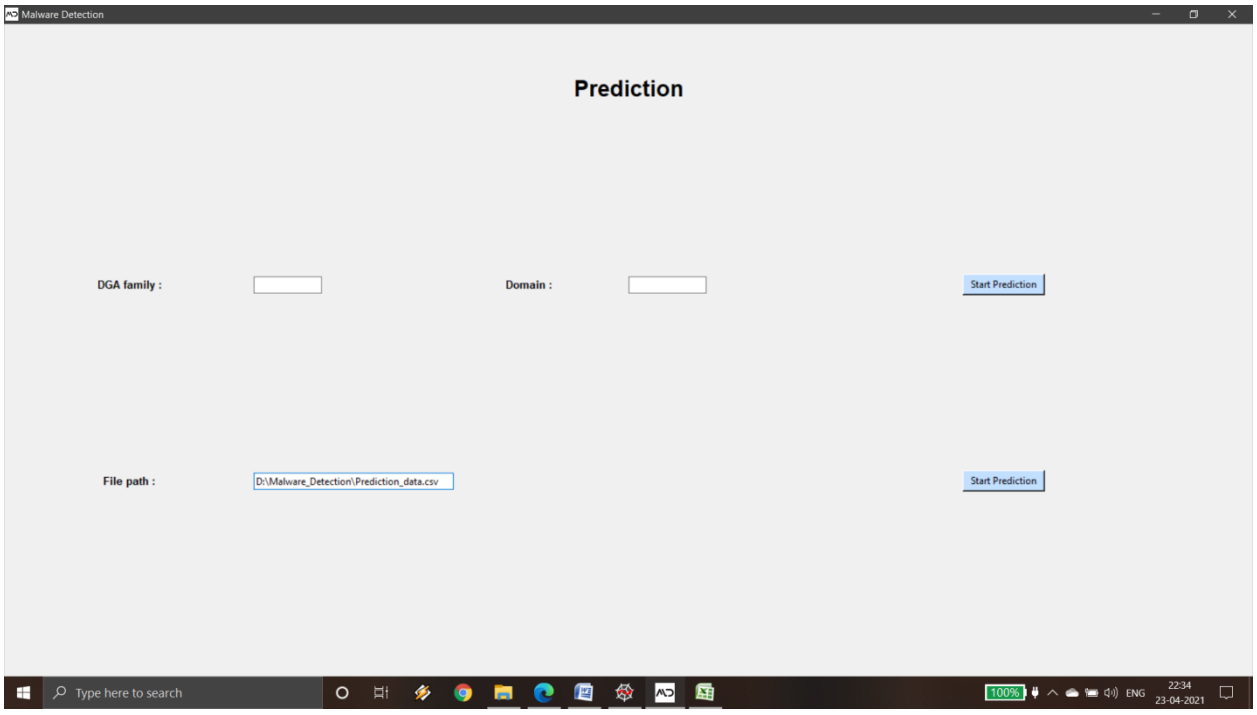

Figure 4.8: Prediction for a list of domains

Result for a list of domains

Here the user would get a text message that the result of the prediction for the given list of domains.

The message would contain the file's path created by the system in the folder where the main program is stored.

The prediction of the list would be stored in a CSV file which the user could open and fetch the results.

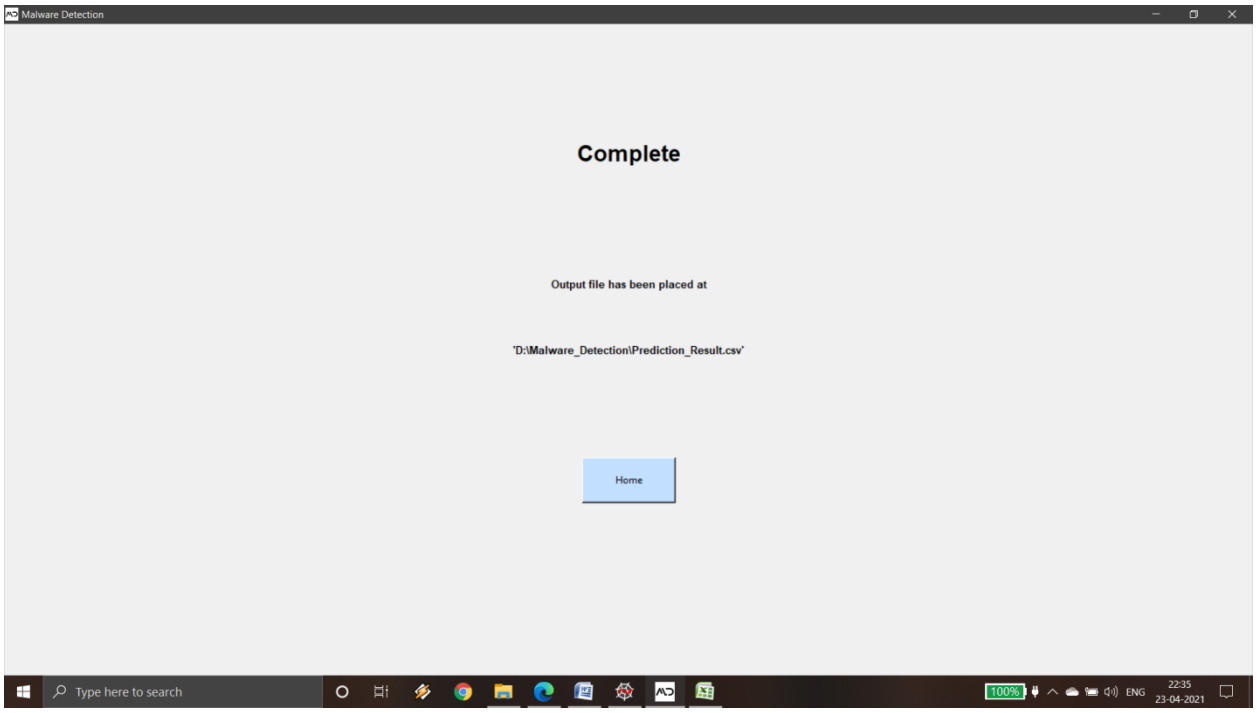

Figure 4.9: File location of result for a list of domains

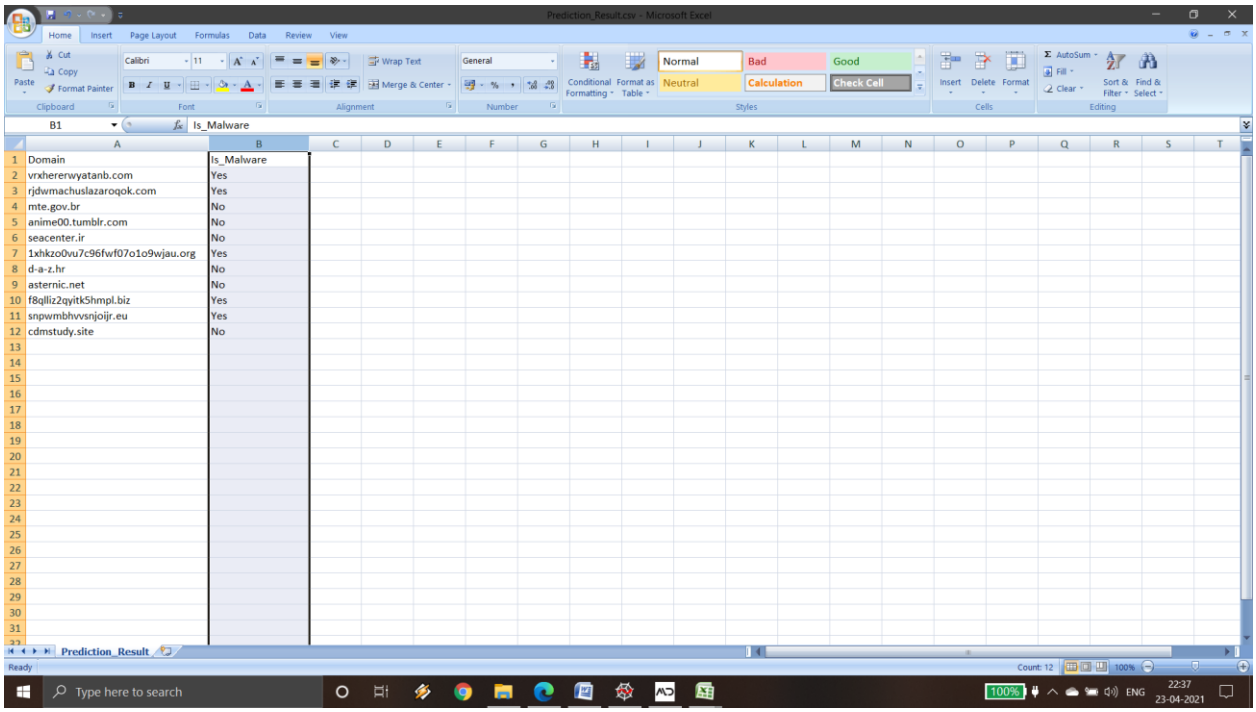

Figure 4.10: CSV file of result for a list of domains.

CHAPTER 5

CONCLUSION & FUTURE WORK

Conclusion

Detecting DGAs is a grand challenge in security areas. Blacklisting is good for handling static methods. However, DGAs are usually used by an attacker to communicate with variety of servers. They are dynamic, so simply using the blacklisting is not sufficient for detecting a DGA. In this research, we have proposed the machine learning framework with the development of a deep learning model to handle DGA threats. The proposed machine learning framework consists of a dynamic blacklist, a feature extractor, a two level machine learning model for classification and clustering, and a prediction model.

We have collected a real-time threat intelligence feed over a one-year period where all domains live threats on the Internet. As the size of the data we collected becomes larger and larger, we have built a deep learning model to perform the classification, which has a better performance than the machine learning algorithms. Based on our extensive experiments on the real-world feed, we have shown that the proposed framework can effectively extract domain name features as well as classify, cluster and detect domain names. We have further used ANN model to improve our classification.

Future Work

In the future, we will further explore deep learning algorithms for domain name clustering and predictions for this research and evaluate them on a real-world. The most common method to detect malicious URLs deployed by many antivirus groups is the blacklist method. Blacklists are essentially a database of URLs that have been confirmed to be malicious inthe past. Scope of this project is useful for it helps to prevent malicious activities in cyber world. It is intended to improve the system performance on the based on dataset. Also use new techniques to get accurate result.

REFERENCES

[1] Yi Li, Kaiqi Xiong, Tommy Chin, Chengbin Hu, “A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection” IEEE Access (Volume: 7), 31 January, 2019

[2] T. Chin, K. Xiong and M. Rahouti, "SDN-based kernel modular countermeasure for intrusion detection", Proc. 13rd EAI Int. Conf. Secur. Privacy Commun. Netw., pp. 270- 290, September, 2019.

[3] S.Mammadli, December 2016, “An SDN based framework for guaranteeing security and performance in information-centric cloud networks,” Procedia Computer Science, pp.495-499.

[4] Xiaojie and D.Huailin, W.Qingfeng, July 2009, “A two-hashing table multiple string pattern matching algorithm,” Tenth International Conference on Information Technology: New Generations (ITNG), IEEE, January 2013.

[5] Jong Young Lee, Jun Young Chang and Eul Gyu Im September 2019, “DNS analysis based malware detection system,” Knowledge Engineering and Applications (ICKEA), IEEE International Conference on IEEE , pp. 193-196.