A Project Report

On

Phishing Detection System Through Hybrid Machine Learning Based on URL

Submitted in Partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

SUBMITTED

By

CHALLA DEEPTHI - 20671A1215

KASUKURTHY MAHENDRA - 20671A1224

K. ARUN KUMAR - 20671A1223

Under the esteemed guidance of

Dr. L. Sridhara Rao

ASSOCIATE PROFESSOR

Department of Information Technology

J.B. Institute of Engineering & Technology

(UGC AUTONOMOUS)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Baskar Nagar, Yenkapally, Moinabad Mandal, R.R. District, Telangana (India)-500075

2023-2024

J.B. INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

(Accredited by NAAC, Permanently Affiliated to JNTUH)

Baskar Nagar,Yenkapally, Moinabad Mandal, R.R. Dist.Telangana(India) -500 075

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report entitled "**PHISHING DETECTION SYSTEM THROUGH HYBRID MACHINE LEARNING BASED ON URL**" being submitted to the Department of Information Technology, J.B. Institute of Engineering and Technology, in accordance with Jawaharlal Nehru Technological University regulations as partial fulfillment required for successful completion of Bachelor of Technology is a recordof bonafide work carried out during the academic year 2023-24 by,

CHALLA DEEPTHI - 20671A1215

KASUKURTHY MAHENDRA- 20671A1224

K. ARUN KUMAR - 20671A1223

Internal Guide

Dr. L. SRIDHARA RAO
ASSOCIATE PROFESSOR

Head of the Department

Dr. S. RAVI KUMAR RAJU
ASSOCIATE PROFESSOR

External Examiner

J.B. INSTITUTE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

(Accredited by NAAC, Permanently Affiliated to JNTUH)

Baskar Nagar,Yenkapally, Moinabad Mandal, R.R. Dist.Telangana(India) -500 075

DEPARTMENT OF INFORMATION TECHNOLOGY

DECLARATION

We hereby certify that the Main Project report entitled **"PHISHING DETECTION SYSTEM THROUGH HYBRID MACHINE LEARNING BASED ON URL"** carried out under the guidance of **Dr. S. RAVI KUMAR RAJU, AssociateProfessor** is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology.** This is a record of bonafide work carried out by us and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Date :

Place : Yenkapally

CHALLA DEEPTHI - 20671A1215

KASUKURTHY MAHENDRA 20671A1224

K. ARUN KUMAR - 20671A1223

ACKNOWLEDGEMENT

At outset we express our gratitude to almighty lord for showering his grace and blessings uponus to complete this Main Project. Although my name appears on the cover of this book, many people had contributed in some form or the other to this project Development. We could not have done this Project without the assistance or support of each of the following.

First of all, we highly indebted to Dr. P. C. KRISHNAMACHARY, Principal for giving us the permission to carry out this Main Project.

we would like to thank **Dr. S.RAVI KUMAR RAJU,** Associate Professor & Head of the Department of INFORMATION TECHNOLOGY**,** for being moral support throughout the period of the study in the Department.

we are grateful to **Dr. L. SRIDHARA RAO**, Assistant Professor of the Department of INFORMATION TECHNOLOGY, for his valuable suggestions and guidance given by him during the of this Project work.

We would like to thank Teaching and Non-Teaching Staff of Department of Information Technology for sharing their knowledge with us.

CHALLA DEEPTHI - 20671A1215

KASUKURTHY MAHENDRA - 20671A1224

K. ARUN KUMAR - 20671A1223

ABSTRACT

Internet worms are one of the most common and most dangerous attacks among cybercrimes. The aim of these attacks is to steal the information used by individuals and organizations to conduct transactions. Internet worm contain various hints among their contents and web browser-based information. The purpose of this study is to perform Extreme Learning Machine (ELM) based classification for 30 features including Phishing Websites Data in UC Irvine Machine Learning Repository database. This study uses machine learning models such as decision tree (DT), linear regression (LR), random forest (RF), naive Bayes (NB), K-neighbors classifier (KNN), support vector classifier (SVC), and proposed hybrid LSD model, which is a combination of logistic regression, support vector machine, and decision tree (LR+SVC+DT) with soft and hard voting, to defend against phishing attacks with high accuracy and efficiency. Cyber security confronts a tremendous challenge of maintaining the confidentiality and integrity of user's private information such as password and PIN code. Billions of users are exposed daily to fake login pages requesting secret information. There are many ways to trick a user to visit a web page such as, phishing mails, tempting advertisements, click-jacking, malware, SQL injection, session hijacking, man-in-the-middle, denial of service and cross-site scripting attacks.

1. INTRODUCTION

Internet use has become an essential part of our daily activities as a result of rapidly growing technology. Due to this rapid growth of technology and intensive use of digital systems, data security of these systems has gained great importance. The primary objective of maintaining security in information technologies is to ensure that necessary precautions are taken against threats and dangers likely to be faced by users during the use of these technologies [1]. Internet worm is defined as imitating reliable websites in order to obtain the proprietary information entered into websites every day for various purposes, such as usernames, passwords and citizenship numbers. Internet worm websites contain various hints
among their contents and web browser-based information [2- 4]. Individual(s) committing the fraud sends the fake website or e-mail information to the target address as if it comes from an organization, bank or any other reliable source that performs reliable transactions.

Contents of the website or the e-mail include requests aiming to lure the individuals to enter or update their personal information or to change their passwords as well as links to websites that look like exact copies of the websites of the organizations concerned . Internet worm Web sites Features Many articles have been published about how to predict the Internet worm websites by using artificial intelligence techniques. We examined Internet worm websites and extracted features of these web sites. Guidelines regarding the extracted features of this database are given below. In the first section we defined rules and we gave equations of web features. We need these equations in order to explain Internet worm attacks characaterization.

2.LITERATURE SURVEY

- **Intelligent rule based Internet worm websites classification**

Internet worm is described as the art of echoing a website of a creditable firm intending to grab user's private information such as usernames, passwords and social security number. Internet worm websites comprise a variety of cues within its content-parts as well as the browser-based security indicators provided along with the website. Several solutions have been proposed to tackle Internet worm. Nevertheless, there is no single magic bullet that can solve this threat radically. One of the promising techniques that can be employed in predicting Internet worm attacks is based on data mining, particularly the `induction of classification rules since anti-Internet worm solutions aim to predict the website class accurately and that exactly matches the data mining classification technique goals. In this study, the authors shed light on the important features that distinguish Internet worm websites from legitimate ones and assess how good rule-based data mining classification techniques are in predicting Internet worm websites and which classification technique is proven to be more reliable.

- **Predicting Internet worm websites based on self-structuring neural network**

we proposed an intelligent model for predicting Internet worm attacks based on artificial neural network particularly self-structuring neural networks. Internet worm is a continuous problem where features significant in determining the type of web pages are constantly changing. Thus, we need to constantly improve the network structure in order to cope with these changes. Our model solves this problem by automating the process of structuring the network and shows high acceptance for noisy data, fault tolerance and high prediction accuracy. Several experiments were conducted in our research, and the number of epochs differs in each experiment. From the results, we find that all produced structures have high generalization ability.

3.SYSTEM ANALYSIS

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of software development methodologies.

3.1 EXISTING SYSTEM:

- Internet worm websites mostly get the e-banking sites and attack their passwords, credit card number, bank account and personal details of the user. He says it's a "New Internet Crime". Comparing with the forma like virus and hacking the Internet worm is mostly popular now days. In this they introduce a risk assessment model with the help of the fuzzy rule and classification algorithm.

3.2 Disadvantages:

- As the social Internet worm attacks underscore the dangers of the public it takes all the personal information's and need to adequate countermeasures.
- In existing methods, they fail to find the Internet worm websites, but they tried it to a mark upto 50% still they can't succeed.

3.3 PROPOSED SYSTEM:

In this study, Extreme Learning Machine (ELM) based classification was performed for the following 30 features extracted based on the features of websites in UC Irvine Machine Learning Repository. Procedural steps for solving the classification problem presented is as follows:

Identification of the problem

This study attempts to solve the problem as to how Internet worm analysis data will be classified.

- **Data set**

Approximately 11,000 data containing the 30 features extracted ased on the features of websites in UC Irvine Machine Learning Repository database.

- **Modeling**

  After the data is ready to be processed, modeling process for the learning algorithm is initiated. The model is basically the construction of the need for output identified in accordance with the task qualifications.

3.4Advantages of Proposed System:

- This study is considered to be an applicable design in automated systems with high performing classification against the Internet worm activity of websites.
- Furthermore, in literature comparisons, this study is observed to be high-performing by having a high performance.

3.5 Software Development Life Cycle:

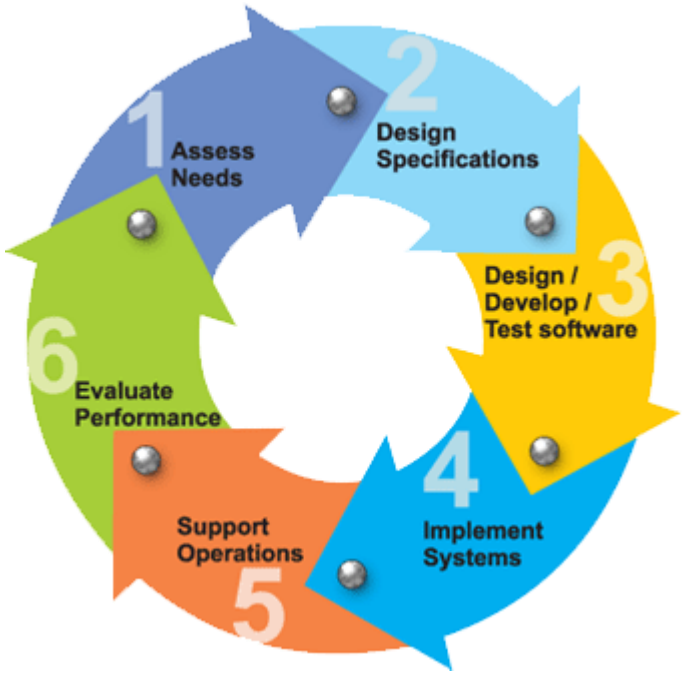

Fig.3.5. Software Development Life Cycle

3.6 Project Implementation Plan:

Requirements:

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements. Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirement gathering phase. This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work. The overall result is the system as a whole and how it performs, not how it is actually going to do it.

Design:

The software system design is produced from the results of the requirements phase. Architects have the ball in their court during this phase and this is the phase in which their focus lies. This is where the details on how the system will work is produced. Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

Implementation:

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle. For a developer, this is the main focus of the life cycle because this is where the code is produced. Implementation my overlap with both the design and testing phases. Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

Testing:

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model. Now let’s delve into some of the traditional and widely used variations.

4.SOFTWARE REQUIREMENT SPECIFICATIONS

4.1 Functional Requirements

The functional requirement refers to the system needs in an exceedingly computer code engineering method.

The key goal of determinant "functional requirements" in an exceedingly product style and implementation is to capture the desired behavior of a software package in terms of practicality and also the technology implementation of the business processes.

4.2 Non-Functional Requirements

All the other requirements which do not form a part of the above specification are categorized as Non-Functional needs. A system perhaps needed to gift the user with a show of the quantity of records during info. If the quantity must be updated in real time, the system architects should make sure that the system is capable of change the displayed record count at intervals associate tolerably short interval of the quantity of records dynamic. Comfortable network information measure may additionally be a non-functional requirement of a system.

The following are the features:

- Accessibility
- Availability
- Backup
- Certification
- Compliance
- Configuration Management
- Documentation
- Disaster Recovery

- Efficiency (resource consumption for given load)
- Interoperability

4.3 Software Requirement Specifications:

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Operating system **:** Windows XP/7/10
- Coding Language **:** Html, JavaScript,
- Development Kit **:** Anaconda prompt
- Programming language: python
- Libraries **:** NLTK

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- System : Pentium IV 2.4 GHz.
- Hard Disk : 100 GB.
- Monitor : 15 VGA Color.
- Mouse **:** Logitech
- RAM : 1GB

5. SYSTEM DESIGN

5.1 System Architecture:

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It's a serious impact on the later part, notably testing and maintenance.

The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

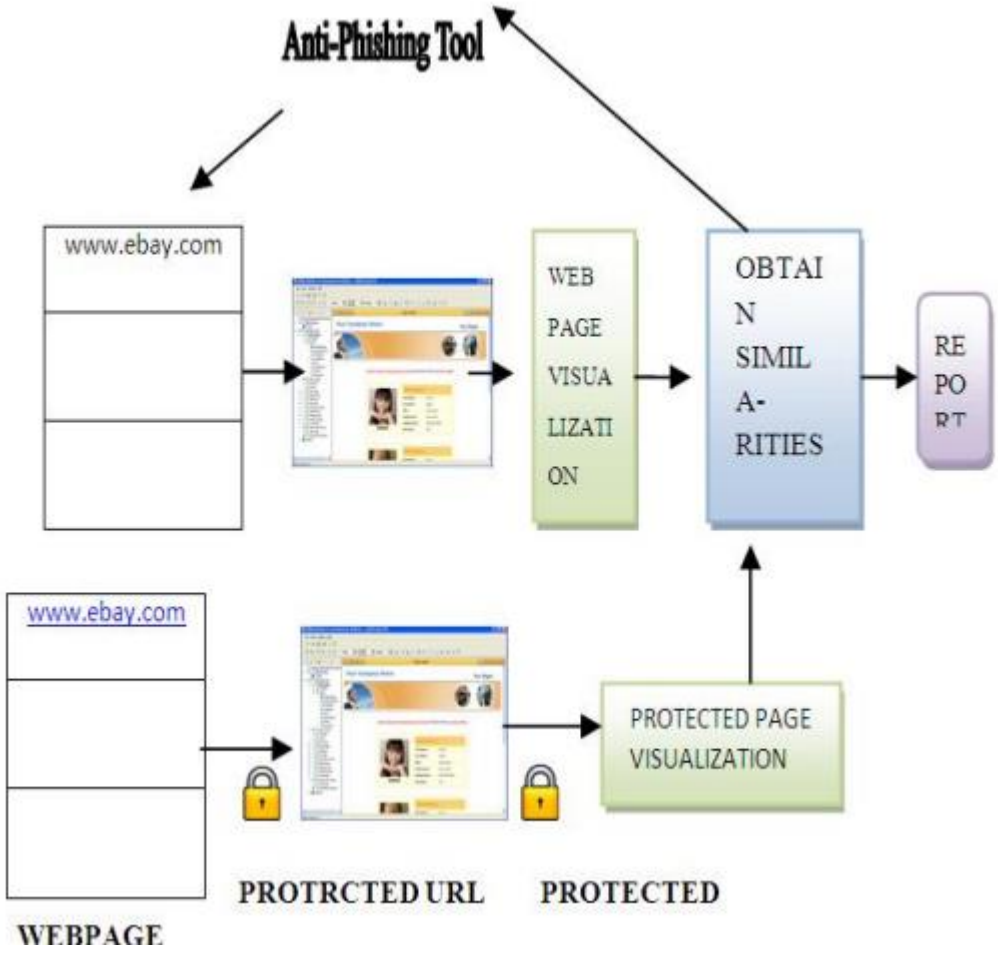

Fig.5.2.System Architecture

5.2 Design Tool Used

There are two broad categories of diagrams and they are again divided into subcategories −

- Structural Diagrams.
- Behavioral Diagrams.

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspectsrepresent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, objects, components, and nodes. The fourstructural diagrams are −

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Class Diagram**:**

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Active class is used in a class diagram to represent the concurrency of the system. Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagramat the time of system construction.

Object Diagram:

Object diagrams can be described as an instance of class diagram. Thus, these diagrams aremore close to real-life scenarios where we implement a system. Object diagrams are a set of objectsand their relationship is just like class diagrams. They also represent the static view of the system.The usage of object diagrams is similar to class diagrams but they are used to build prototype of asystem from a practical perspective.

Component Diagram:

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system arranged in different groups depending upon their relationship. Now, these groups are known as components. Finally, it can be said component diagrams are usedto visualize implementation During the design phase, software artifacts (classes, interfaces, etc.) of a system.

Deployment Diagram:

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

Note − If the above descriptions and usages are observed carefully then it is very clear that all thediagrams have some relationship with one another Again, the deployment diagram is dependent upon the components, which are used to make component diagrams.

Behavioral Diagrams:

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered.

behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. UML has the following five types of behavioral diagrams

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram.

Use case diagram:

Use case diagrams are a set of use cases, actors, and their relationships. They represent theuse case view of a system. A use case represents a particular functionality of a system. Hence, usecase diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

Sequence Diagram:

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Collaboration Diagram:

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objectsand links. The purpose of the collaboration diagram is similar to a sequence diagram. However, the specific purpose of the collaboration diagram is to visualize the organization of objects and their interaction.

State chart Diagram:

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to representthe event driven state change of a system. It basically describes the state change of a class, interface,etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

Activity Diagram:

Activity diagram describes the flow of control in a system. It consists of activities and links.The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

5.3 UML Diagram

5.3.1 Use Case Diagram

The use case diagram is used to show what the user/actor does with the system. They are used to depict the functional requirements of the system and its interaction with the users (actors).A use case is a representation of various scenarios. Use case diagrams give us a view of how the entire system works without going into the implementation details.

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running/operation. So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagrams are one of them. Now as we have to discuss that theuse case diagram is dynamic in nature there should be some internal or external factors for makingthe interaction.

These internal and external agents are known as actors. So use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of anapplication. A single use case diagram captures a particular functionality of a system. So to modelthe entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal andexternal influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. In brief, thepurposes of use case diagrams can be as follows:

a. Used to gather requirements of a system.

b. Used to get an outside view of a system.

c. Identify external and internal factors influencing the system.

d. Show the interacting among the requirements are actor

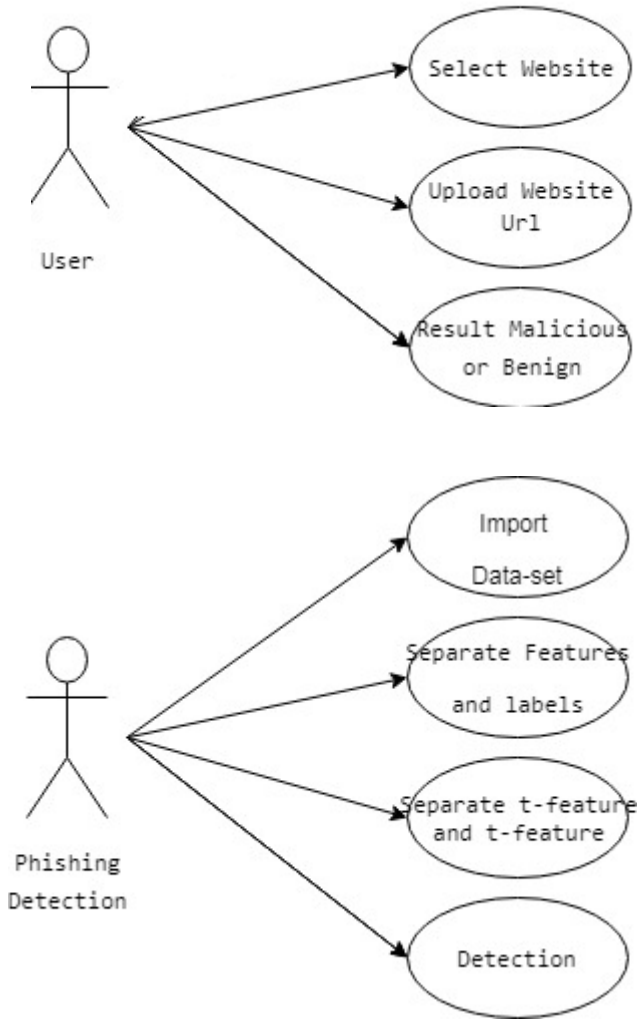

Fig.5.3.1.Use case Diagram

5.3.2 Class Diagram

Class diagrams are the main building blocks of every object-oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is basedon OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objectsand the static relationship in between them.

The main purpose to use class diagrams are:

1. The upper part holds the name of the class
2. The middle part contains the attributes of the class
3. The bottom part gives the methods or operations the class can take or undertake

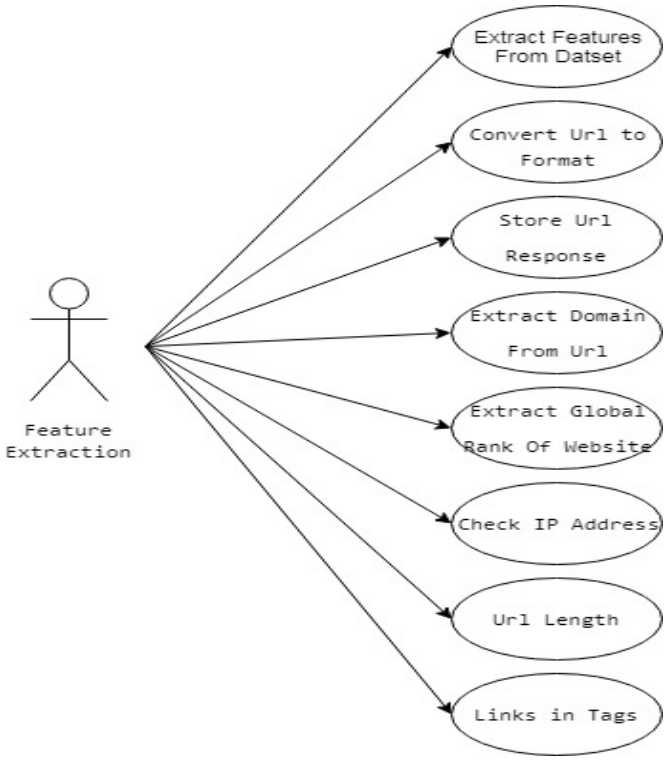

Fig.5.3.2.Class Diagram

5.3.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a newsystem.

The aim of a sequence diagram is to define event sequences, which would have a desired outcome. The focus is more on the order in which messages occur than on the message per user. However, the majority of sequence diagrams will communicate what messages are sent and the order in which they tend to occur.

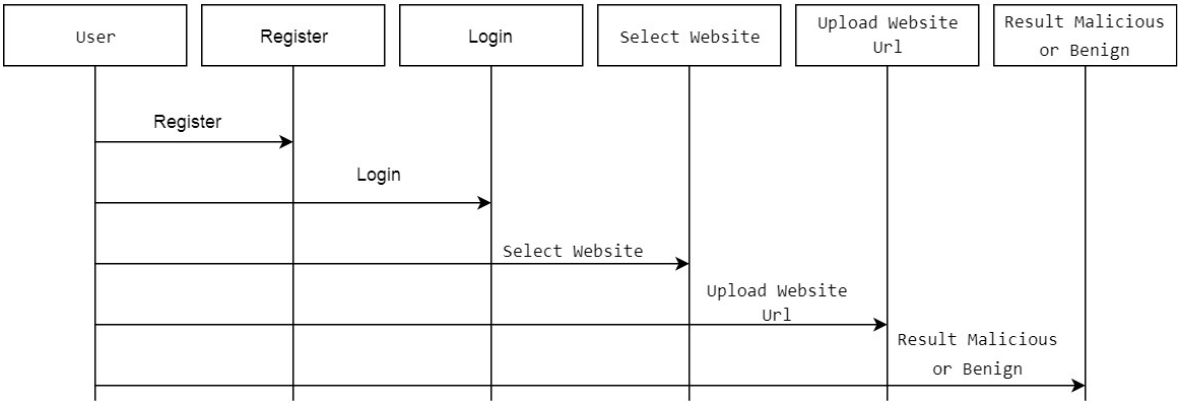

Fig.5.3.3.1 Sequence Diagram

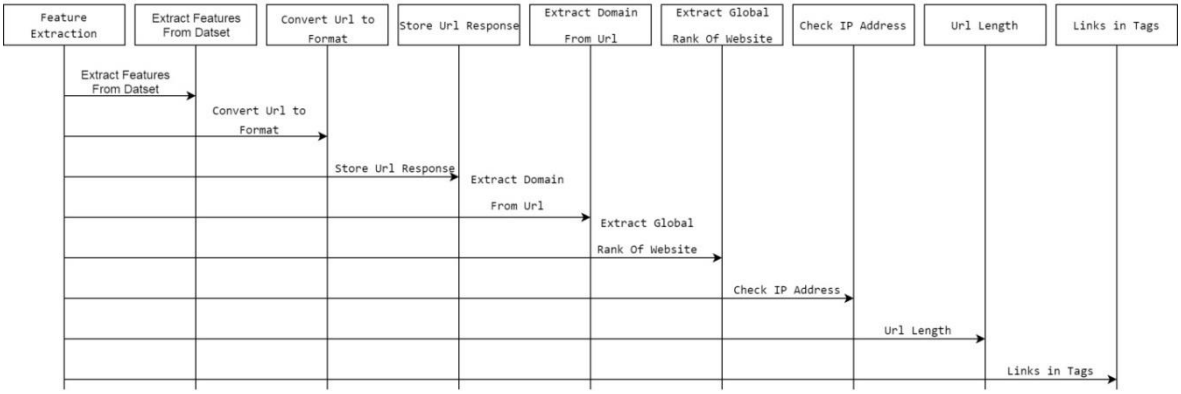

Fig.5.3.3.2 Sequence Diagram

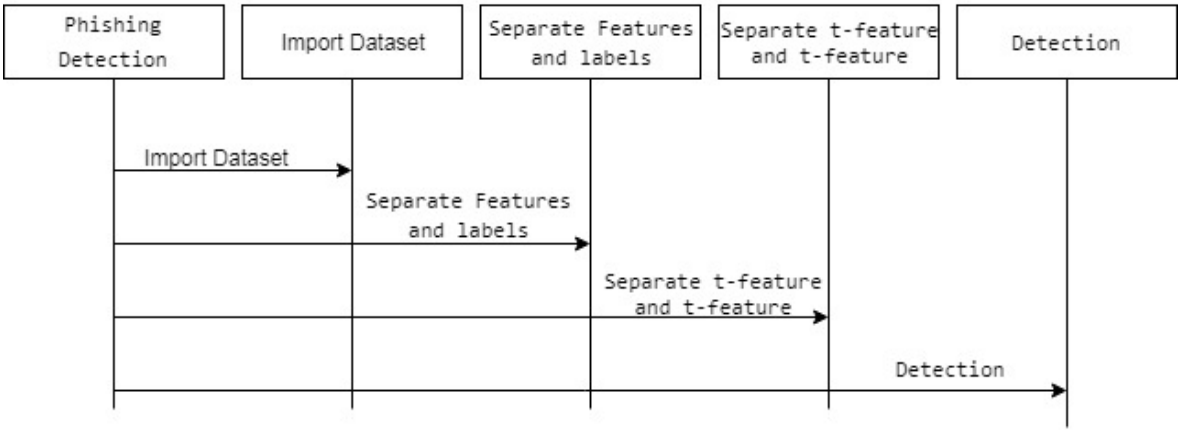

Fig.5.3.3.2 Sequence Diagram

5.3.4 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

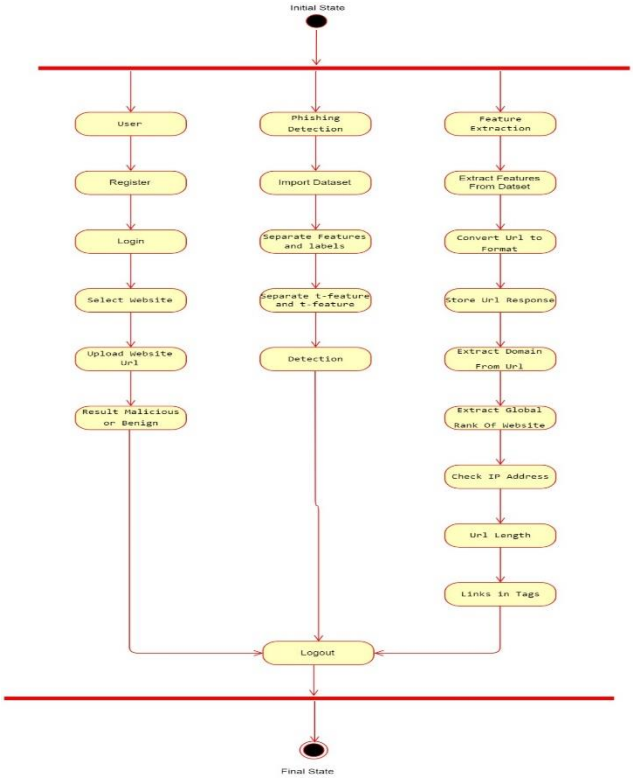

Fig.5.3.4. Activity Diagram

5.3.5 Collaboration Diagram

Collaboration diagrams are nothing different from sequence diagrams. Sequence diagramsshow the flow of the system whereas the Collaboration diagrams show how objects interact with each other in the system. They are used to depict the object behavior of the system. Collaboration diagrams are also known as communication diagrams.

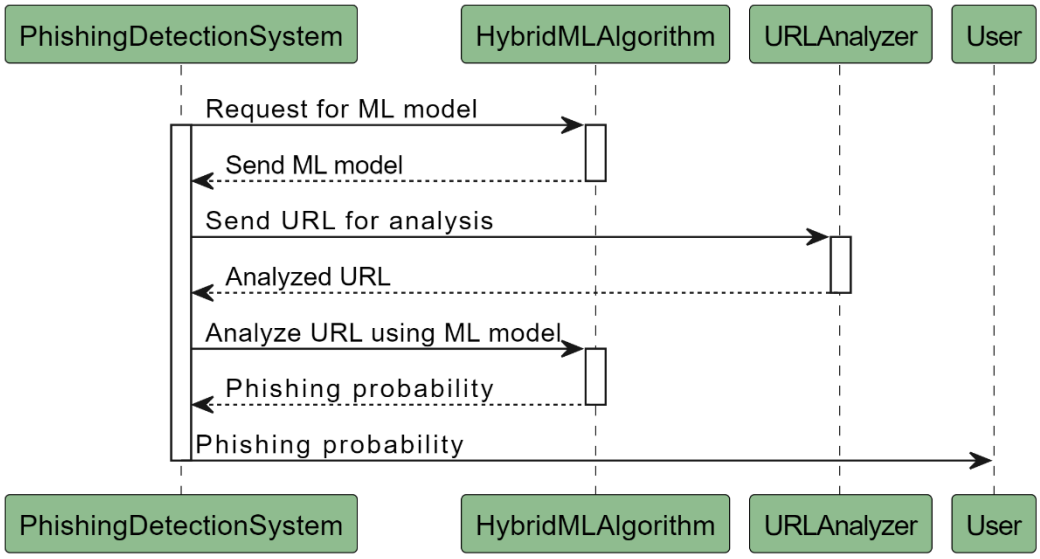

Fig.5.3.5.Collaboration diagram

5.3.6 Entity-Relationship Diagram

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

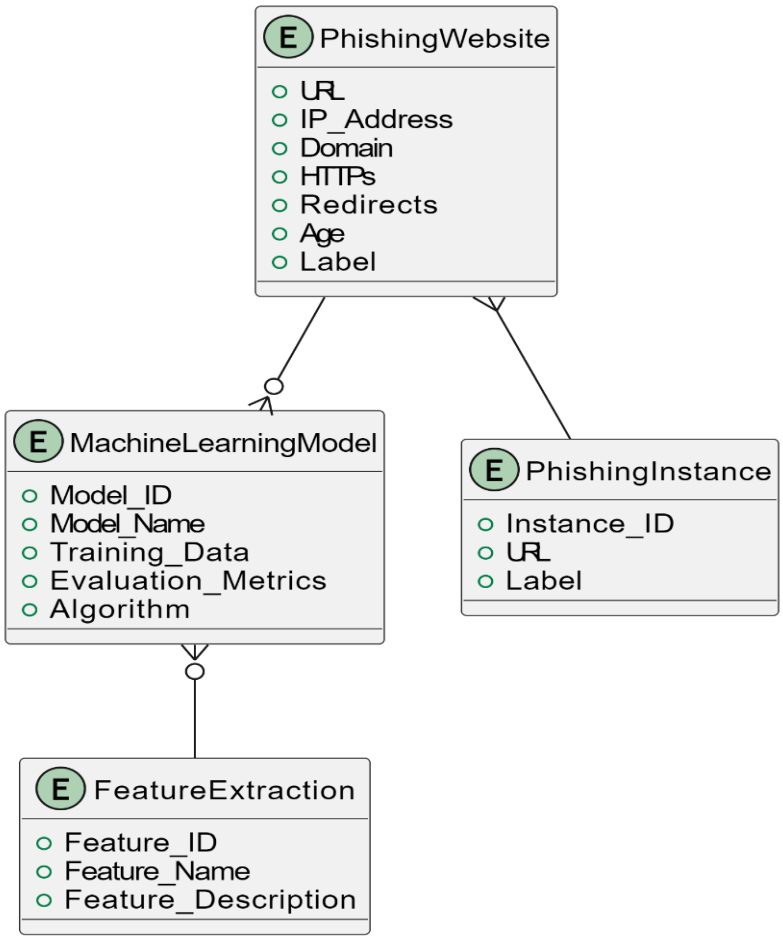

Fig.5.3.6. ER diagram

6.IMPLEMENTATION

6.1 Introduction

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable:

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable:

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor:

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program

Script: Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program: The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

6.2 Technologies Used

Python

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

• Open-source general-purpose language.
• Object Oriented, Procedural, Functional
• Easy to interface with C/Obj C/Java/Fortran
• Easy-Ish to interface with C++ (via SWIG)
• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

6.2 Technologies Used

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String

- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python

type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function**.**You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There

are three types of functions in python:help(),min(),print().

Python Namespace: Generally speaking, a namespace (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace. This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs. Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

6.3 coding Standards:

Data collection:

- we will take Phishing URL data set from Kaggle which has features as tweet data and labels as Phishing URL or not.

Data preprocessing:

- Features are extracted from data set and stored in variable as xtrain variable and labels are stored in y train variable. Data is preprocessing by standard scalar function and new features and labels are generated.

Testing training:

- In this stage data is sent to testing and training function and divided in to four parts x test train, and y test train. Train variables are used for passing to algorithm whereas test are used for calculating accuracy of the algorithm.

Initializing Multiple Algorithms and training with Logistic regression:

- In this stage machine learning algorithms are initialized and train values are given to algorithm by this information algorithm will know what are features and what are labels. Then data is modeled and stored as pickle file in the system which can be used for prediction.
- Data set is trained with multiple algorithms and accuracy of each model is calculated and best model is used for prediction

Predict data:

- In this stage new data is taken as input and trained models are loaded using pickle and then values are preprocessed and passed to predict function to find out result which is showed on web application.

6.4 Modules:

1. Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.

2. Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won't use any other.

3. wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

8. NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.

14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

15. Scapy. A packet sniffer and analyzer for python made in python.

16. pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that. Do check it out.

18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or

SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.
- Django vs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the deploymentsection to make it accessible on the web.

Python-Data Base Communication

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs

Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQL database. This is an introduction into using SQLite and MySQL from Python. The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces. The DB-API has been defined as a common interface, which can be used to access relational databases. In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface. SQLite is a simple relational

database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM. It was developed for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well.

If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object. The connection object will represent the database. The argument of connection - in the following example "company db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.
For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.
MySQL Connector/Python includes support for:

- Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
- Converting parameter values back and forth between Python and MySQL data types, for example Python determined MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance
- All MySQL extensions to standard SQL syntax.
- Protocol compression, which enables compressing the data stream between the client and server.
- Connections using TCP/IP sockets and on Unix using Unix sockets.
- Secure TCP/IP connections using SSL.

- Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library.

6.5 Unit Test Cases:

- As indicated above, code is usually developed in a file using an editor.
- To test the code, import it into a Python session and try to run it.
- Usually there is an error, so you go back to the file, make a correction, and test again.
- This process is repeated until you are satisfied that the code works.
- The entire process is known as the development cycle.
- There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.
- This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

7. SYSTEM TESTING

7.1 TEST PLAN

SYSTEM TESTING:

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated.Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

MODULE TESTING:

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

INTEGRATION TESTING:

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

ACCEPTANCE TESTING:

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

7.2 Test Planning:

Test Data Collection and Preparation:

Gather a comprehensive dataset of both legitimate and phishing URLs. Ensure the dataset covers a diverse range of URL patterns, including newly emerging phishing techniques. Split the dataset into training, validation, and test sets.

Model Training and Validation:

Train the hybrid machine learning model using the URL-based features. Evaluate the model's performance on the validation set and fine-tune the hyperparameters to optimize accuracy, precision, recall, and F1-score.

Model Testing:

Thoroughly test the trained model on the held-out test set to assess its real-world performance. Evaluate the model's ability to detect both known and unknown (zero-day) phishing attacks.

Robustness Testing:

Assess the model's resilience to adversarial attacks, where attackers may try to bypass the detection system by obfuscating or manipulating the URL features. Conduct stress tests to ensure the model maintains high performance under such conditions.

Scalability and Performance Testing:

Evaluate the system's ability to handle large volumes of URLs in a real-time environment. Measure the processing time, throughput, and resource utilization to ensure the system can meet the required performance targets.

Usability and Integration Testing:

Ensure the phishing detection system integrates seamlessly with other security components, such as web browsers or email clients. Test the user experience and provide clear feedback to users about the detected threats.

Continuous Monitoring and Updating:

Implement a process to continuously monitor the performance of the phishing detection system, update the model with new data, and adapt to evolving phishing techniques.

Reporting and Documentation:

Maintain detailed test plans, test cases, and test results to ensure the system's reliability and facilitate future maintenance and improvements.

8.5 TEST CASES:

Phish Id	**Test Case Name**	**Test Case Desc.**	**Test Steps**			**Test Case Status**	**Test Passing**
			Step	**Expected**	**Actual**		
1	Upload the tasks dataset	Verify either file is loaded or not	If dataset is not uploaded	It cannot display the file loaded message	File is loaded which displays task waiting time	High	High
2	Upload patients dataset	Verify either dataset loaded or not	If dataset is not uploaded	It cannot display dataset reading process completed	It can display dataset reading process completed	low	High
3	Preprocessing	Whether preprocessing on the	If not applied	It cannot display the necessary	It can display the necessary	Medium	High

		dataset applied or not		data for further process	data for further process		
4	Prediction Random Forest	Whether Prediction algorithm applied on the data or not	If not applied	Random tree is not generated	Random tree is generated	High	High
5	Recommendation	Whether predicted data is displayed or not	If not displayed	It cannot view prediction containing patient data	It can view prediction containing patient data	High	High

7.4 bug report

No Bugs Found.

8. RESULT SCREEN

8.1 Execution Steps:

Step 1: Open anaconda prompt from search.

Step 2: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter)

Step 3: type python app.py (enter u will get ip address ctrl + click on link website will open)

How to run jupyter file.

Step4: Open Another anaconda prompt from search.

Step 5: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter).

Step 6: Type jupyter notebook Phishing URL Detection.ipynb (enter).

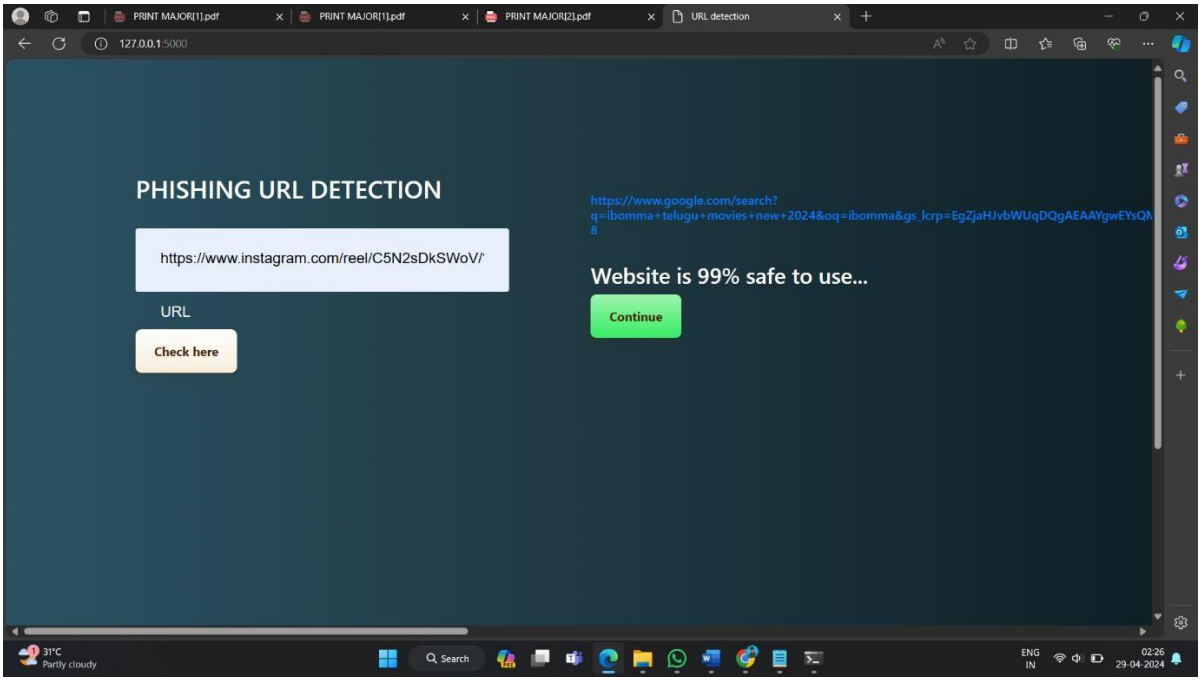

Fig.8.1.1.Result Screen

Case2:

Step 1: Open anaconda prompt from search.

Step 2: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter)

Step 3: type python app.py (enter u will get ip address ctrl + click on link website will open)

How to run jupyter file.

Step4: Open Another anaconda prompt from search.

Step 5: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter).

Step 6: Type jupyter notebook Phishing URL Detection.ipynb (enter).

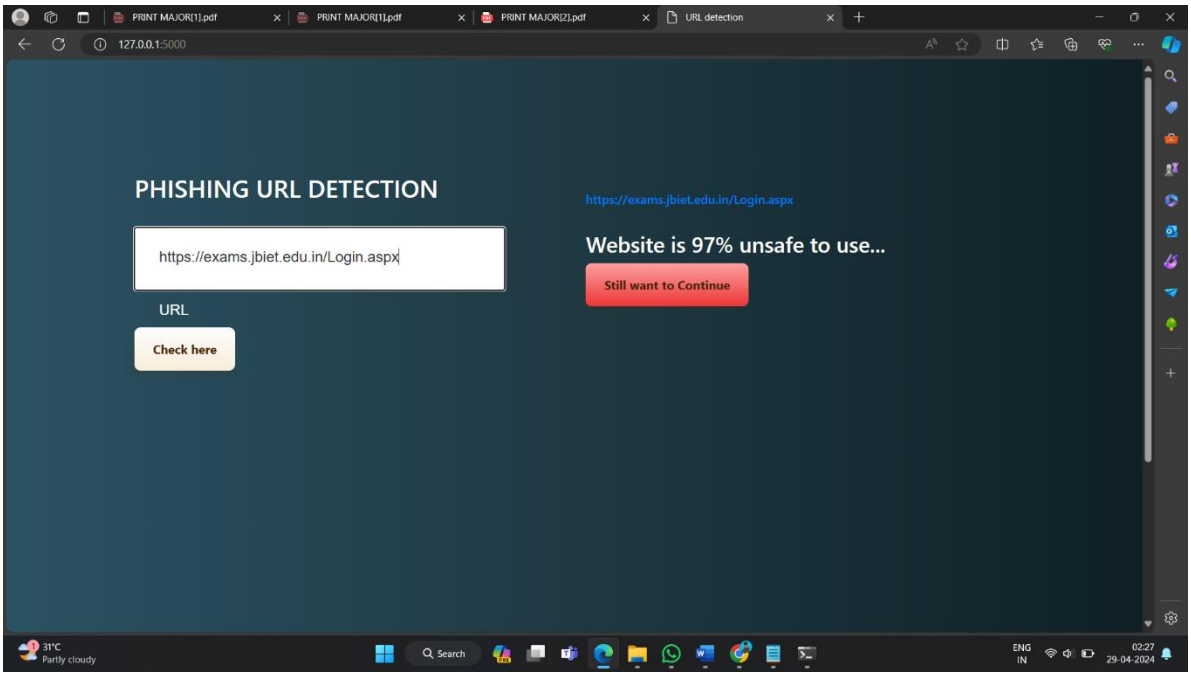

Fig.8.1.2.Result Screen

Case 3:

Step 1: Open anaconda prompt from search.

Step 2: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter)

Step 3: type python app.py (enter u will get ip address ctrl + click on link website will open)

How to run jupyter file.

Step4: Open Another anaconda prompt from search.

Step 5: Type cd C:\Users\kasuk\OneDrive\Desktop\Phishing-URL-Detection-hybride (enter).

Step 6: Type jupyter notebook Phishing URL Detection.ipynb (enter).

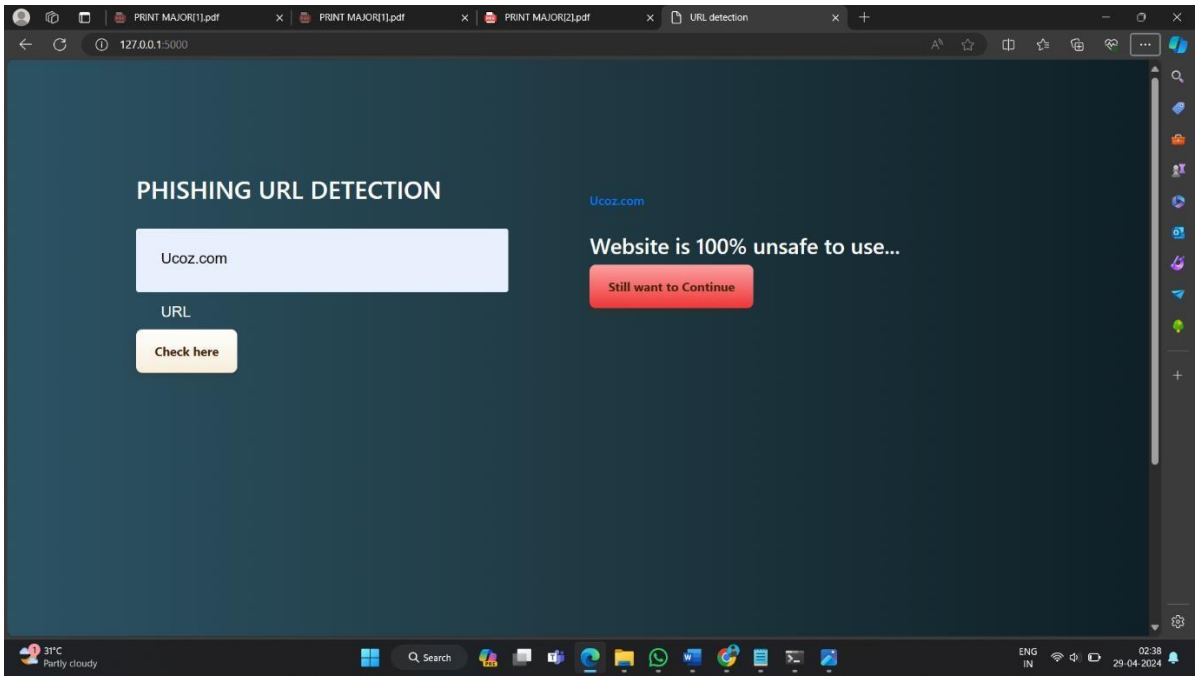

Fig.8.1.3.Result Screen

9. CONCLUSION AND FUTURE ENHANCMENT

In conclusion, we defined features of Internet worm attack and we proposed a classification model in order to classification of the Internet worm attacks. This method consists of feature extraction from websites and classification section. In the feature extraction, we have clearly defined rules of Internet worm feature extraction and these rules have been used for obtaining features. In order to classification of these feature, SVM, NB and ELM were used. In the ELM, 6 different activation functions were used and ELM achieved highest accuracy score.

Future Work:

We will explore further correlations between Internet worm sites and hosting and DNS registration companies. We will also look at additional features that can be leveraged, such as Content Security Policies, certificate authorities, and TLS fingerprinting. Additionally, we will implement other machine learning techniques such as random forest classifiers for speed and accuracy and compare them to SVMs, and neural networks. Finally, we will look at the underlying HTML structure for features, specifically counts of tags, placement of tags, use of and counts of specific JavaScript functions, inline and included CSS, etc.

10. BIBLIOGRAPHY

References

[1] G. Canbek and “A Review on Information, Information Security and Security Processes,” Politek. Derg., vol. 9, no. 3, pp. 165–174, 2006.

[2] L. McCluskey, F. Thabtah, and R. M. Mohammad, “Intelligent rulebased Internet worm websites classification,” IET Inf. Secur., vol. 8, no. 3, pp. 153–160, 2014.

[3] R. M. Mohammad, F. Thabtah, and L. McCluskey, “Predicting Internet worm websites based on self-structuring neural network,” *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443–458, 2014.

[4] R. M. Mohammad, F. Thabtah, and L. McCluskey, “An assessment of features related to Internet worm websites using an automated technique,” *Internet Technol. …*, pp. 492–497, 2012.

[5] W. Hadi, F. Aburub, and S. Alhawari, “A new fast associative classification algorithm for detecting Internet worm websites,” Appl. Soft Compute. J., vol. 48, pp. 729–734, 2016.

[6] N. Abdelhamid, “Multi-label rules for Internet worm classification,” Appl. Comput. Informatics, vol. 11, no. 1, pp. 29–46, 2015.

[7] N. Sanglerdsinlapachai and A. Rungsawang, “Using domain top-page similarity feature in machine learning-based web Internet worm detection,” in 3rd International Conference on Knowledge Discovery and DataMining, WKDD 2010, 2010, pp. 187–190.]

11.APPENDIXES

SAMPLE CODE:

#importing required libraries

```
import numpy as np
import pandas as pd
import matplotlib. Pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

#Loading data into data frame

```
data = pd.read_csv("phishing.csv")
data.head()
```

#Shape of datagram

```
data.shape
#Listing the features of the dataset

data. columns
```

app.py:

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction
```

```
file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()


app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template ('index.html',xx =round(y_pro_non_phishing,2),url=url )
    return render_template ("index.html", xx =-1)


if __name__ == "__main__":
    app.run(debug=True)
```

feature.py:

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
```

```
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass

        try:
            self.urlparse = urlparse(url)
            self.domain = self.urlparse.netloc
        except:
            pass

        try:
            self.whois_response = whois.whois(self.domain)
        except:
            pass
```

```
self.features.append(self.UsingIp())
self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())


self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
self.features.append(self.LinksPointingToPage())
self.features.append(self.StatsReport())
```

```
    # 1.UsingIp
    def UsingIp(self):
        try:
            ipaddress.ip_address(self.url)
            return -1
        except:
            return 1

    # 2.longUrl
    def longUrl(self):
        if len(self.url) < 54:
            return 1
        if len(self.url) >= 54 and len(self.url) <= 75:
            return 0
        return -1

    # 3.shortUrl
    def shortUrl(self):
        match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                  'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                  'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                  'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.
im|link\.zip\.net', self.url)
        if match:
            return -1
        return 1

    # 4.Symbol@
    def symbol(self):
```

```
    if re.findall("@",self.url):
        return -1
    return 1

# 5.Redirecting//
def redirecting(self):
    if self.url.rfind('//')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1

# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
```

```
        except:
            return 1

    # 9.DomainRegLen
    def DomainRegLen(self):
        try:
            expiration_date = self.whois_response.expiration_date
            creation_date = self.whois_response.creation_date
            try:
                if(len(expiration_date)):
                    expiration_date = expiration_date[0]
            except:
                pass
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

            age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-
creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1

    # 10. Favicon
    def Favicon(self):
        try:
            for head in self.soup.find_all('head'):
                for head.link in self.soup.find_all('link', href=True):
                    dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                    if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                        return 1
            return -1
        except:
```

```
        return -1

# 11. NonStdPort
def NonStdPort(self):
    try:
        port = self.domain.split(":")
        if len(port)>1:
            return -1
        return 1
    except:
        return -1

# 12. HTTPSDomainURL
def HTTPSDomainURL(self):
    try:
        if 'https' in self.domain:
            return -1
        return 1
    except:
        return -1

# 13. RequestURL
def RequestURL(self):
    try:
        for img in self.soup.find_all('img', src=True):
            dots = [x.start(0) for x in re.finditer('\.', img['src'])]
            if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                success = success + 1
            i = i+1

        for audio in self.soup.find_all('audio', src=True):
            dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
            if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
                success = success + 1
            i = i+1
```

```
            for embed in self.soup.find_all('embed', src=True):
                dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
                if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for iframe in self.soup.find_all('iframe', src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

    # 14. AnchorURL
    def AnchorURL(self):
        try:
            i,unsafe = 0,0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not
(url in a['href'] or self.domain in a['href']):
                    unsafe = unsafe + 1
                i = i + 1

            try:
                percentage = unsafe / float(i) * 100
```

```
            if percentage < 31.0:
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
                return 0
            else:
                return -1
        except:
            return -1

    except:
        return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
    try:
        i,success = 0,0

        for link in self.soup.find_all('link', href=True):
            dots = [x.start(0) for x in re.finditer('\.', link['href'])]
            if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
                success = success + 1
            i = i+1

        for script in self.soup.find_all('script', src=True):
            dots = [x.start(0) for x in re.finditer('\.', script['src'])]
            if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
                success = success + 1
            i = i+1

        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                return 1
            elif((percentage >= 17.0) and (percentage < 81.0)):
                return 0
            else:
```

```
            return -1
        except:
            return 0
    except:
        return -1

# 16. ServerFormHandler
def ServerFormHandler(self):
    try:
        if len(self.soup.find_all('form', action=True))==0:
            return 1
        else :
            for form in self.soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank":
                    return -1
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
                else:
                    return 1
    except:
        return -1

# 17. InfoEmail
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?]", self.soap):
            return -1
        else:
            return 1
    except:
        return -1

# 18. AbnormalURL
def AbnormalURL(self):
    try:
        if self.response.text == self.whois_response:
```

```
            return 1
        else:
            return -1
    except:
        return -1

# 19. WebsiteForwarding
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
         return -1

# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
```

```
        return -1

# 22. UsingPopupWindow
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

# 23. IframeRedirection
def IframeRedirection(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

# 24. AgeofDomain
def AgeofDomain(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass

        today  = date.today()
        age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
        if age >=6:
            return 1
```

```
            return -1
        except:
            return -1

    # 25. DNSRecording
    def DNSRecording(self):
        try:
            creation_date = self.whois_response.creation_date
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

            today  = date.today()
            age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
            if age >=6:
                return 1
            return -1
        except:
            return -1

    # 26. WebsiteTraffic
    def WebsiteTraffic(self):
        try:
            rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" +
url).read(), "xml").find("REACH")['RANK']
            if (int(rank) < 100000):
                return 1
            return 0
        except :
            return -1

    # 27. PageRank
    def PageRank(self):
        try:
```

```
        prank_checker_response = requests.post("https://www.checkpagerank.net/index.php",
{"name": self.domain})

        global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
        if global_rank > 0 and global_rank < 100000:
            return 1
        return -1
    except:
        return -1


# 28. GoogleIndex
def GoogleIndex(self):
    try:
        site = search(self.url, 5)
        if site:
            return 1
        else:
            return -1
    except:
        return 1

# 29. LinksPointingToPage
def LinksPointingToPage(self):
    try:
        number_of_links = len(re.findall(r"<a href=", self.response.text))
        if number_of_links == 0:
            return 1
        elif number_of_links <= 2:
            return 0
        else:
            return -1
    except:
        return -1

# 30. StatsReport
def StatsReport(self):
```

```
        try:
            url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.l
y', url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.
158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.1
51\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'

'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.2
24\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19
\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.
183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.1
9|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1


    def getFeaturesList(self):
        return self.features
```