PHISHING WEBSITE DETECTION USING MACHINE LEARNING

A Project report submitted in partial fulfillment of requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By

K. ASRITH KUMAR REDDY (209X1A0552)

B. HEMANTH (209X1A05J7)

B. JAYA LINGESWARA REDDY (209X1A0536)

Under the esteemed guidance of

Sri. M. ANAND
Assistant Professor
Department of C.S.E.

Department of Computer Science and Engineering

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

2023 – 2024

Department of Computer Science and Engineering

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

CERTIFICATE

This is to certify that the Project Work entitled 'Phishing Website Detection Using Machine Learning' *is a bonafide record of work carried out by*

K. ASRITH KUMAR REDDY (209X1A0552)

B. HEMANTH (209X1A05J7)

B. JAYA LINGESWARA REDDY (209X1A0536)

Under my guidance and supervision in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

……..

Sri. M. Anand
Assistant Professor,
Department of C.S.E.,
G. Pulla Reddy Engineering College,
Kurnool.

……..

Dr. N. Kasiviswanath
Professor & Head of the Department,
Department of C.S.E.,
G. Pulla Reddy Engineering College,
Kurnool.

DECLARATION

We hereby declare that the project titled "**PHISHING WEBSITE DETECTION USING MACHINE LEARNING**" is an authentic work carried out by us as the students of **G. PULLA REDDY ENGINEERING COLLEGE(Autonomous) Kurnool,** during 2023-24 and has not been submitted elsewhere for the award of any degree or diploma in part or in full to any institute.

K. Asrith Kumar Reddy
(209X1A0552)

B. Hemanth
(209X1A05J7)

B. Jaya Lingeswara Reddy
(209X1A0536)

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to our project guide **Sri. M. Anand**, **Assistant Professor** of Computer Science and Engineering Department, G. Pulla Reddy Engineering College, for his immaculate guidance, constant encouragement and cooperation which have made possible to bring out this project work.

We are grateful to our project incharge **Dr. C. Sreedhar**, **Professor and Associate HOD** of Computer Science and Engineering Department, G. Pulla Reddy Engineering College, for helping us and giving us the required information needed for our project work.

We are thankful to our Head of the Department **Dr. N. Kasiviswanath**, for his whole hearted support and encouragement during the project sessions.

We are grateful to our respected Principal **Dr. B. Sreenivasa Reddy** for providing requisite facilities and helping us in providing such a good environment.

We wish to convey our acknowledgements to all the staff members of the Computer Science and Engineering Department for giving the required information needed for our project work.

Finally, we wish to thank all our friends and well wishers who have helped us directly or indirectly during the course of this project work.

ABSTRACT

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. By extracting and comparing different characteristics between legitimate and phishing URLs, the suggested method uses gradient boosting classifier to identify phishing URLs. The studies' findings demonstrate that the suggested approach successfully identifies legitimate websites from bogus ones in real time.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

Phishing is a fraudulent technique that uses social and technological tricks to steal customer identification and financial credentials. In our daily life, we carry out most of our work on digital platforms. Using a computer and the internet in many areas facilitates our business and private life. It allows us to complete our transaction and operations quickly in areas such as trade, health, education, communication, banking, aviation, research, engineering, entertainment, and public services. The users who need to access a local network have been able to easily connect to the Internet anywhere and anytime with the development of mobile and wireless technologies. Although this situation provides great convenience, it has revealed serious deficits in terms of information security. Thus, the need for users in cyberspace to take measures against possible cyber-attacks has emerged. These attacks are mainly targeted in the following areas: fraud, forgery, force, shakedown, hacking, service blocking, malware applications, illegal digital contents and social engineering. According to Kaspersky's data, the average cost of an attack in 2019 (depending on the size of the attack) is between $ 108K and $ 1.4 billion. In addition, the money spent on global security products and services is around $ 124 billion. Among these attacks, the most widespread and also critical one is “phishing attacks”. It causes pecuniary loss and intangible damages.

In United States businesses, there is a loss of US$2billion per year because their clients become victim to phishing. In 3rd Microsoft Computing Safer Index Report released in February 2014, it was estimated that the annual worldwide impact of phishing could be as high as $5 billion. Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them but it is very important to enhance phishing detection techniques. The general method to detect phishing websites by updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also known as “blacklist" method. To evade blacklists attackers uses creative techniques to fool users by modifying the URL to appear legitimate via obfuscation and many other simple techniques including: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic generation of new URLs etc.

Major drawback of this method is that, it cannot detect zero hour phishing attack. Heuristic based detection which includes characteristics that are found to exist in phishing attacks in reality and can detect zero-hour phishing attack, but the characteristics are not guaranteed to always exist in such attacks and false positive rate in detection is very high. To overcome the drawbacks of blacklist and heuristic-based method, many security researchers now focused on machine learning techniques. Machine learning technology consists of many algorithms which requires past data to make a present model to detect future results.

In order to receive confidential data, criminals develop unauthorized replicas of a real website and email, typically from a financial institution or other organization dealing with financial data. This e-mail is rendered using a legitimate company's logos and slogans. The design and structure of HTML allow copying of images or an entire website. Also, it is one of the factors for the rapid growth of Internet as a communication medium, and enables the misuse of brands, trademarks and other company identifiers that customers rely on as authentication mechanisms. To trap users, Phisher sends "spooled" mails to as many people as possible. When these e-mails are opened, the customers tend to be diverted from the legitimate entity to a spoofed website.

Malicious Web sites largely promote the growth of Internet criminal activities and constrain the development of Web services. As a result, there has been strong motivation to develop systemic solution to stopping the user from visiting such Web sites.
URLs of the websites are separated into 3 classes:

Benign: Safe websites with normal services

Spam: Website performs the act of attempting to flood the user with advertising or sites such as fake surveys and online dating etc.

Malware: Website created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems.

There is a significant chance of exploitation of user information. For these reasons, phishing in modern society is highly urgent, challenging, and overly critical. The method of reaching target users in phishing attacks has continuously increased since the last decade. This method has been carried out in the 1990s as an algorithm-based, in the early 2000s based on e- mail, then as Domain Spoofing and in recent years via HTTPs. Due to the size of the mass attacked in recent years, the cost and effect of the attacks on the users have been high.

The average financial cost of the data breach as part of the phishing attacks in 2019 is $ 3.86 million, and the approximate cost of the BEC (Business Email Compromise) phrases is estimated to be around $12 billion. Also, it is known that about 15% of people who are attacked are at least one more target. With this result, it can be said that phishing attacks will continue to being carried out in the ongoing years. So, we proposed a system with the help of machine learning techniques and algorithms like Logistic Regression, KNN, SVC, Random Forest, Decision Tree, XGB Classifier and Naïve Bayes to predict Phishing Website based on different parameters like extracted by the website link entered by the user in the front end.

1.2 PROBLEM STATEMENT

1. The Cyber-attacks are growing faster than usual rate, it became evident that necessary steps should be taken in-order to get them under control. Among various cyber-attacks, Phishing websites is one of the popular and commonly used attack to steal users personal information and financial information by manipulating the website URL and IP addresses.
2. The main focus in this project is to implement the better model for detecting these phishing websites using ML algorithms.

1.3 EXISTING SYSTEM

"Phishing Detection Using Machine Learning": This paper proposes an approach of phishing detection system to detect blacklisted URL also known as phishing websites, so that individual can be alerted while browsing or accessing a particular website. Therefore, it can be utilized for identification and authentication and become a legitimate tool to prevent an individual from getting tricked. The system fosters many features in comparison of other software. Its unique features such as capturing blacklisted URL's from the browser directly to verify the validity of the website, notifying user on blacklisted websites while they are trying to access through popup, and also notifying through email. This system will assist user to be alert when they are trying to access a blacklisted website.

1.3.1 LIMITATIONS

i. As a list of blacklisted URLs is being used to detect the accuracy of predicting the correct results may be very low. There isn't any standard list of URLs published by any standard organization.

ii. As a part of daily life, we may encounter many new URL's which seems to be same as original ones and not present in the list. It is hard to differentiate URL's.

1.4 PROPOSED SYSTEM

As part of our project, we are implementing a new framework to detect these phished websites using Machine Learning Algorithms. As we aren't using any list of URLs, this can be used to detect any new URL that the user encounters. We use URL features as parameters to the model. There are 30 features that are being considered as major decision parameters. Using these 30 features an ML model is made to meet the challenges in existing system.

1.4.1 ADVANTAGES

The major advantages of this project are:

- We can identify fake websites and helps users safe from exploitation of their information.
- We can stop unsafe transactions identifying these fake websites.
- We can add this approach to any browser to make it secure browser.

1.5 SIGNIFICANCE AND RELEVANCE OF WORK

a. In today's society, as the phishing attacks have become evident, the need of counteractions are necessary.
b. The main problem of Phishing has raised abundantly in the last 3 years due to Covid –
c. 19. Number of Cases encountering on a daily basis has increased by 125% in 2021 from 2020. If this is the scenario happening, the phishing cases may raise up to 500 per day by 2025.
d. Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced.
e. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database.
f. Therefore, we are developing a website to predict the phishing website URL's which helps the society to cut down these attacks and save themselves from the frauds.

1.6 OBJECTIVES

a. To develop a novel approach to detect malicious URL and alert users.
b. To apply ML techniques in the proposed approach in order to analyze the real time URLs and produce effective results.
c. Using zero - hour mechanism to cut-down Phishing attacks.

1.7 METHODOLOGY

a. To Take a dataset and divide it into two parts training and testing datasets in different ratios.
b. Implement a fine approach to detect phishing attacks using various machine learning algorithms.
c. The algorithm which gives the better accuracy rate comparatively is taken as our final prediction model along with the lexical features combined.

1.8 ORGANIZATION OF THE REPORT

Chapter 1

Overview

The overview provides the basic layout and the insight about the work proposed. It briefs the entire need of the currently proposed work.

Problem statement

A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. We have identified the gap between addressed or a condition to be improved upon. We have identified the gap between the current existing work and the desired work. Considering the needs of society, we have made an effort to approach the existing problems.

Significance and Relevance of Work

We have mentioned about the contribution of our work to the society.

Objectives

A project objective describes the desired results of the work. We have mentioned about the work we are trying to accomplish in this section.

Methodology

A methodology is a collection of methods, practices, processes and techniques. We have explained in this section about the working of the project in a brief way.

Chapter 2

Literature Survey: the purpose of a literature review is to gain an understanding of the existing resources to a particular topic or area of study. We have referred to many research papers relevant to our work in a better way.

Chapter 3

System Requirements: This section contains the hardware software requirements that are needed to implement the project.

Chapter 4

System Analysis: System Analysis is a document that describes about the existing system and proposed system in the project. And also describes about advantages and disadvantages in the project.

Chapter 5

System design: System design is a document that describes about the project modules, Activity diagram, Use Case Diagram, Data Flow Diagram, and Sequence Diagram detailed in the project.

Chapter 6

Implementation: Implementation is a document that describes about the detailed concepts of the project. Also describes about the algorithm with their detailed steps. And also, about the codes for implementation of the algorithm.

Chapter 7

Testing

Testing is a document that describes about the

a. **Methods of testing**
   This contains information about Unit testing, Validation testing, Functional testing, Integration testing, User Acceptance testing.
b. **Test Cases**
   In Test Cases we contain the detailed description about program Testcases.

Chapter 8

Conclusion and Future Enhancement: Conclusion and Future Enhancement is a document that describes about the brief summary of the project and undetermined events that will occur in that time.

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 A MACHINE LEARNING APPROACH FOR PHISHING ATTACK DETECTION

Authors: M. M. Vilas, K. P. Ghansham, S. P. Jaypralash and P. Shila

Phishing is one kind of cyber-attack and at once, it is a most dangerous and common attack to acquire personal information, account details, credit card details, organizational details or password of a user to conduct transactions. Phishing websites seem to like the appropriate ones and it is difficult to differentiate among those websites. The motive from that study is to perform ELM derived from different 30 main components which are categorized using the ML approach. Most of the phishing URLs use HTTPS to avoid getting detected. There are three ways for the detection of website phishing. The primitive approach evaluates different items of URL, the second approach analyzing the authority of a website and calculating whether the website is introduced or not and it also analyzing who is supervising it, the third approach checking the genuineness of the website.

Remarks: Contending approaches are fallen with less accuracy when compared with APE.

2.2 DETECTING PHISHING WEBSITES USING MACHINE LEARNING

Authors: Alswailem, B. Alabdullah, N. Alrumayh and A. Alsedrani

In this paper, we offer an intelligent system for detecting phishing websites. The system acts as an additional functionality to an internet browser as an extension that automatically notifies the user when it detects a phishing website. The system is based on a machine learning method, particularly supervised learning. We have selected the Random Forest technique due to its good performance in classification. Our focus is to pursue a higher performance classifier by studying the features of phishing website and choose the better combination of them to train the classifier. As a result, we conclude our paper with good accuracy and combination of 26 features.

Remarks: Low ranging values reduces accuracy and increases execution time.

2.3 MACHINE LEARNING BASED PHISHING DETECTION FROM URLS

Authors: M. Korkmaz, O. K. Sahingoz and B. Diri

In this paper, they proposed a machine learning-based phishing detection system by using eight different algorithms to analyze the URLs, and three different datasets to compare the results with other works. The experimental results depict that the proposed models have an outstanding performance with a success rate. In this paper, we aimed to implement a phishing detection system by using some machine learning algorithms. The proposed systems are tested with some recent datasets in the literature and reached results are compared with the newest works in the literature. The comparison results show that the proposed systems enhance the efficiency of phishing detection and reach very good accuracy rates. As future works, firstly, it is aimed to create a new and huge dataset for URL based Phishing Detection Systems. With the use of this dataset, we plan to enhance our system by using some hybrid algorithms, and also deep learning mode.

Remarks: Implementing with many algorithms and with various datasets may provide better results compared to previous works but lexical features of an URL are not taken into consideration which provides optimal results with better accuracy rates.

2.4 DETECTION AND PREVENTION OF PHISHING WEBSITES USING MACHINELEARNING APPROACH

Authors: V.Patil, P.Thakkar, C.Shah, T.Bhat, and S P Godse

In this paper, the author has discussed three approaches for detecting phishing websites. First is by analyzing various features of URL, second is by checking legitimacy of website by knowing where the website is being hosted and who are managing it, and the third approach is visual appearance based analysis for checking genuineness of website. The authors have used Machine Learning techniques and algorithms for evaluation of these different features of URL and websites.

Remarks: A particular challenge in this domain is that criminals are constantly making new strategies to counter our defense measures. To succeed in this context, we need algorithms that continually adapt to new examples and features of phishing URL's.

SYSTEM SPECIFICATIONS

3. SYSTEM REQUIREMENTS AND SPECIFICATION

3.1 HARDWARE REQUIREMENTS

Hardware requirements refer to the essential physical components and specifications necessary for the proper functioning of a software application. These include the type and speed of the CPU, RAM and the capacity and speed of the storage devices, whether hard drives or solid- state drives. Graphics processing units (GPUs) may be specified for applications with graphical demands. These specifications guide users and system administrators in configuring and maintaining the hardware environment for seamless software operation

1. Operating System : Microsoft Windows 10/8/7
2. Hard Disk : 500 GB.
3. Ram : 4 GB.
4. Processor : Minimum Core i5
5. Any desktop / Laptop system with above configuration or higher level

3.2 SOFTWARE REQUIREMENTS

Software requirements encompass the essential elements necessary for the development, implementation, and functioning of a software system. These typically include the specification of programming languages, frameworks, and libraries required for development, as well as the need for specific databases or data storage solutions. Overall, software requirements serve as a comprehensive guideline, outlining the technological, functional, and operational prerequisites vital for the successful deployment and performance of a software application.

1. Operating system: Windows XP / 7
2. Coding Language: Python, HTML
3. Version: Python 3.6.8
4. IDE: Python 3.6.8 IDE
5. ML Packages: NumPy, Pandas, sci-kit learn, Matplotlib, Seaborn, Flask, PymySql,
6. ML Libraries: whois, xgboost, favicon, beautiful soup, googlesearch.
7. ML Algorithms: Logistic Regression, Random Forest Classifier, K-Nearest neighbor, Artificial Neural Networks and XGB Classifier.
8. Other Requirements: Notepad, XAMPP Control Panel

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 INTRODUCTION

System analysis is a thorough examination of the current software or the business processes that the software aims to address. The goal is to understand the intricacies, identify challenges, and define opportunities for improvement. Analysts gather information from stakeholders, existing documentation, and through direct observations to comprehend the software's functionalities, user requirements, and the broader context in which it operates.

During system analysis in a software project, the emphasis is on defining clear and comprehensive requirements. This includes understanding user needs, business rules, and any constraints that might impact the design and development of the software. Analysts aim to bridge the gap between the current state of the software and what is needed for it to align effectively with organizational goals and user expectations.

The insights gained from system analysis guide subsequent phases of the software development life cycle. This includes system design, coding, testing, and implementation. The thorough understanding of user requirements and system functionalities obtained during the analysis phase helps in creating a software solution that is not only technically sound but also addresses the practical needs and challenges faced by end-users and the organization. In essence, system analysis in a software project serves as the cornerstone for making informed decisions and laying the groundwork for successful software development and implementation.

Furthermore, in a software project, system analysis involves assessing the feasibility of the proposed changes or developments. This includes evaluating the technical, operational, and economic aspects to determine the practicality and viability of implementing the suggested solutions. System analysis sets the stage for effective project planning, resource allocation, and risk management, ensuring that the subsequent phases of the software development life cycle proceed with a well-defined understanding of the system's requirements and the strategic objectives of the project.

4.2 FEASIBILITY STUDY

A feasibility study is a crucial preliminary step in the planning and initiation of a project, providing an in-depth assessment of the practicality, viability, and potential success of the proposed project. The primary objective is to determine whether the project is worth pursuing by evaluating various aspects such as technical feasibility, operational feasibility, economic feasibility, legal feasibility, and scheduling feasibility. The findings of a feasibility study provide stakeholders, including project managers, investors, and decision-makers, with valuable insights to make informed decisions about whether to proceed with the project. If the study reveals that the project is not feasible, stakeholders can save resources by avoiding investments in an endeavor that may not yield the desired outcomes. Conversely, if the study indicates feasibility, it serves as a foundation for detailed project planning and execution.

4.2.1 TECHNICAL FEASIBILITY

Technical feasibility, a key component of the broader feasibility study, is a comprehensive evaluation that focuses on the technological aspects of a proposed project. This assessment is critical in determining whether the envisaged solution aligns with the existing technology infrastructure and can be effectively implemented within the technical capabilities of the organization. In the initial stages of technical feasibility analysis, there is a thorough examination of the required technology. This involves assessing the availability and suitability of essential components, such as hardware, software, and any specialized tools or systems necessary for project execution. This step ensures that the proposed project's technological needs can be met either through existing resources or by acquiring or developing new technology.

An essential aspect of technical feasibility is evaluating the technical skills and expertise available within the organization. This entails assessing the competency of the development team, understanding whether additional training is required, or if there is a need to augment the team with individuals possessing specific technical skills. A well-equipped and skilled team is fundamental to overcoming technical challenges and ensuring the successful implementation of the project.

Risk analysis is inherent in technical feasibility, aiming to identify and mitigate potential obstacles that could hinder project success. This includes assessing risks related to technology obsolescence, security vulnerabilities, or dependencies on external systems. Proactive identification and mitigation of these risks contribute to a more robust and resilient technical framework.

Scalability and flexibility are also critical considerations in technical feasibility. The assessment revolves around determining whether the proposed solution can adapt to future growth and changes in requirements without necessitating substantial modifications or disruptions. A scalable and flexible technological infrastructure is essential for accommodating the evolving needs of the organization.

4.2.2 OPERATIONAL FEASIBILITY

Operational feasibility is a crucial aspect of a feasibility study that assesses the practicality and viability of implementing a proposed project within an organization's existing operational environment. This evaluation focuses on whether the proposed solution can seamlessly integrate into the daily business operations and whether it is acceptable and adaptable for end-users. The primary goal is to ensure that the project aligns with the organizational processes and can be effectively utilized by stakeholders without causing disruptions.

Key considerations in operational feasibility include user acceptance, training requirements, and potential changes in roles and responsibilities. Evaluating whether the proposed project can be seamlessly integrated into the day-to-day activities of the organization ensures that the implementation process is smooth and does not adversely affect productivity. A positive operational feasibility assessment indicates that the proposed solution is not only technically and economically viable but is also practical and operationally sound within the context of the organization's current operational landscape.

4.2.3 ECONOMIC FEASIBILITY

Economic feasibility is a critical component of a feasibility study that evaluates the financial viability and potential economic benefits of a proposed project. This assessment involves a thorough analysis of the costs associated with project development and implementation against the expected economic returns and benefits. The primary objective is to determine whether the project is financially justifiable and aligns with the organization's budgetary constraints.

In the economic feasibility analysis, various costs are considered, including development costs, operational costs, maintenance costs, and any other expenditures associated with the project's lifecycle. These costs are compared to the anticipated benefits, which may include revenue generation, cost savings, increased efficiency, or other economic advantages. This comparison helps stakeholders understand the financial implications of undertaking the project and whether the expected benefits outweigh the incurred costs.

4.3 DATASET

The dataset used in this machine learning project was obtained from Kaggle, a well-known platform for data science competitions and datasets. 11430 URLs with 30 retrieved characteristics are part of the supplied dataset. The dataset is intended to serve as the benchmark for phishing detection systems that employ machine learning. The collection comprises precisely 45% genuine URLs and 55% phishing URLs. Each instance contains 30 features. Each feature is associated with a rule. If the rule satisfies, it is termed as phishing. If the rule doesn‘t satisfy then it is termed as legitimate. The features take three discrete values. 1 if the rule is satisfied, 0 if the rule is partially satisfied and -1 if the rule is not satisfied. For this research implementation, this dataset is used since it is the most recent dataset accessible in the public domain.

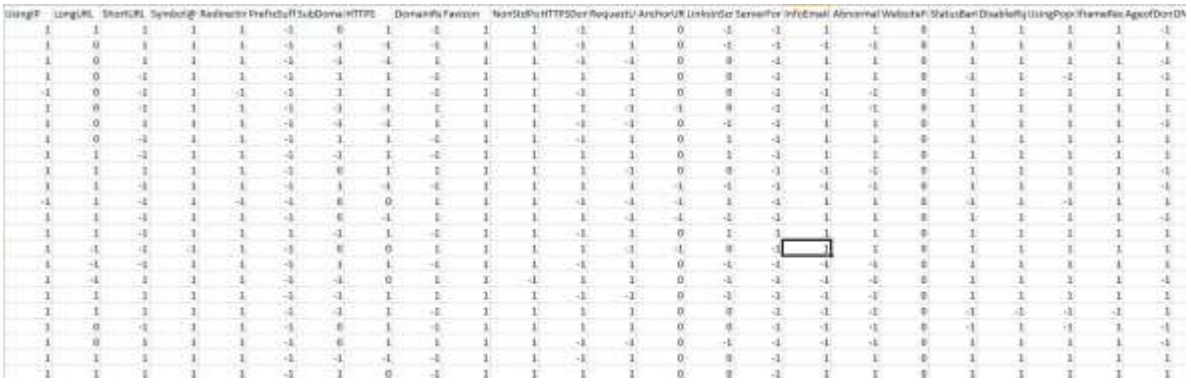

Fig 4.3 Sample of dataset

4.3.1 FEATURES OF DATASET

The dataset contains several factors that should be considered when deciding whether a website URL is licit or phishing. The factors for discovery and bracket of phishing websites are as follows

- Address Bar based features
- Abnormal based features
- HTML and JavaScript based features
- Domain based features

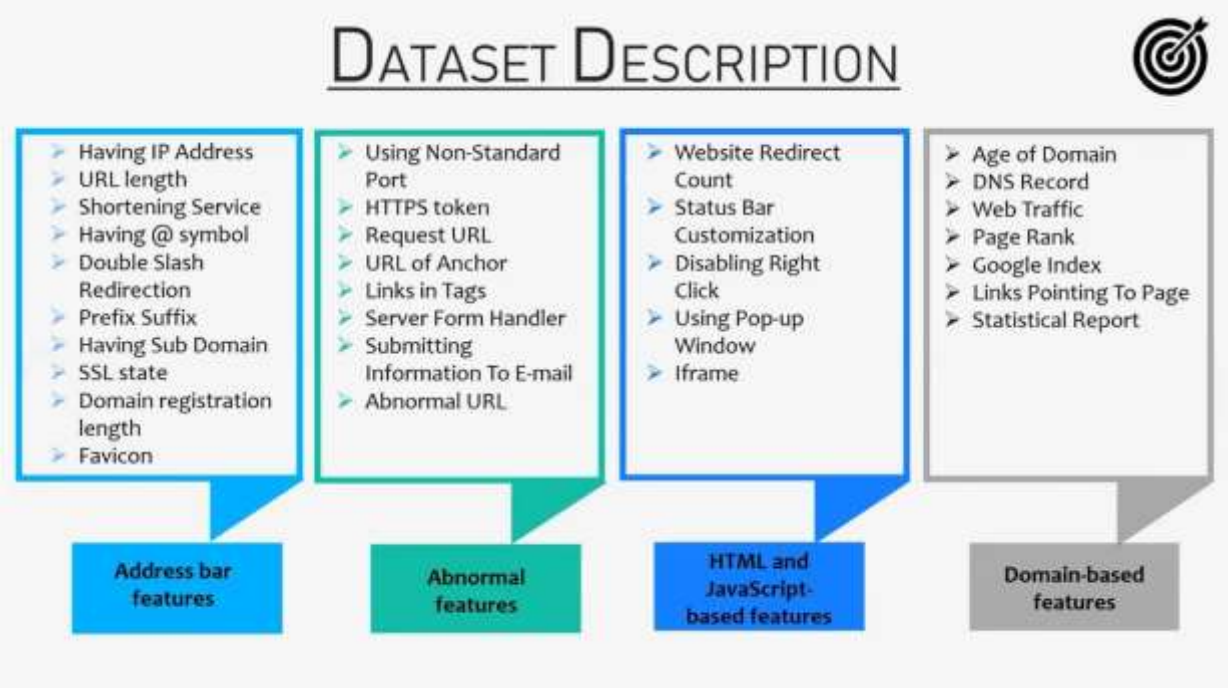

Fig 4.3.1(a) Classification of Features of Dataset

Address Bar Based Features

1. **Having an IP Address**

   If an IP address is used in the URL instead of the domain name, such as http://217.102.24.235/sample.html.

2. **Length of URL**

   Phishers may conceal the suspicious element of the URL in the address bar by using a lengthy URL.

3. **URL Shortening Service**

   Provides access to a website with a lengthy URL. The URL http://sharif.hud.ac.uk/, for example, may be abbreviated to bit.ly/1sSEGTB.

4. **Using the @ sambol**

   @ in the URL causes the browser to disregard anything before the @ symbol, and the true address often follows the @ symbol.

5. **Double Slash Redirection**

   The presence of / in a URL indicates that the user will be redirected to another website.

6. **Prefix Suffix**

   Phishers often add prefixes or suffixes separated by (-) to domain names in order to give visitors the impression that they are dealing with a reputable website.

7. **Using a Subdomain**

   Using a subdomain in the URL.

8. **SSL Status**

   Indicates whether or not a website employs SSL.

9. **Domain Registration Length**

   Because a phishing website only exists for a brief time,

10. **Favicon**

    A favicon is a visual image (icon) that is connected with a particular website. If the favicon is loaded from a domain different than the one displayed in the address bar, the site is most certainly a Phishing attempt.

11. **Using Non-Standard Ports**

    It is much preferable to just open the ports that you need to regulate invasions. Several firewalls, proxy servers, and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only allow access to those that are explicitly allowed.

12. **HTTPS token**

    Using a deceptive https token in the URL. For instance, http://www.mellat-phish.ir

Abnormal Based Features

13. **Request URL**

    The request URL checks to see whether the external items on a site, like as photos, videos, and music, are loaded from another domain.

14 Anchor URL

An anchor is an element specified by the a > tag. This feature is processed in the same way as Request URL.

15. **Links in Tags**

    Websites often utilize Meta tags to provide information about the HTML content, Script tags to generate a client-side script, and Link tags to fetch external online resources. If the domain name in SFHs differs from the domain name of the site.

16. **Server from Handler (SFH)**

    Because process should be performed upon that reported file, SFHs that contains an empty string or roughly clean are suspicious.

17. **Submitting Information Via E-mail**

    A phisher may reroute the user's information to his email address.

18. Incorrect URL

It is derived from the WHOIS database. Identity is usually included in the URL of a legitimate website.

HTML And JavaScript Based Features

19. Website Redirect Count

If the redirection occurs more than four times, it is considered excessive.

20. Status Bar Customization

Utilize JavaScript to display a bogus URL to users in the status bar.

21. Disabling Right Click

This is the same as using on Mouse Over to conceal the Link.

22. Using Pop-up Window

Demonstrating the presence of pop-up windows on the website.

23. IFrame

An IFrame is an HTML element that displays an extra website inside the one that is now shown.

Domain Based Features

24. Domain Age

If the domain is less than a month old.

25. DNS Record

Possessing a DNS record

26. Web Traffic

The number of visits to a website is used to determine its popularity.

27. Page Rank

Page rank is a number that ranges from 0 to 1. PageRank attempts to determine the importance of a site on the Internet.

28. Google Index

This function determines whether or not a website is included in Google's index.

29. Number of Links Pointing To Page

The number of links that point to the web page.

30. Statistical Report

Determine whether or not the IP address belongs to the top phishing IP addresses.

Attribute number	Attributes	Possible values
1	having_IP_Address	-1,1
2	URL_Length	1,0,-1
3	Shortening_Service	1,-1
4	having_At_Symbol	1,-1
5	double_slash_redirecting	-1,1
6	Prefix_Suffix	-1,1
7	having_Sub_Domain	-1,0,1
8	SSLfinal_State	-1,1,0
9	Domain_registration_length	-1,1
10	Favicon	1,-1
11	Port	1,-1
12	HTTPS_token	-1,1
13	Request_URL	-1,1
14	URL_of_Anchor	-1,0,1
15	Links_in_tags	1,-1,0
16	SFH (server form handler)	-1,1,0
17	Submitting_to_email	-1,1
18	Abnormal_URL	-1,1
19	Redirect page	0,1
20	onMouseOver (using to hide link)	1,-1
21	RightClick	1,-1
22	Using pop-up widnow	1,-1
23	Iframe	1,-1
24	age_of_domain	-1,1
25	DNSRecord	-1,1
26	web_traffic	-1,0,1
27	Page_Rank	-1,1
28	Google_Index	-1,1
29	Links_pointing_to_page	1,0,-1
30	Statistical_report	-1,1
Class	Result	-1,1

Fig 4.3.1(b) Features of Dataset

4.4 REQUIREMENTS SPECIFICATION

4.4.1 FUNCTIONAL REQUIREMENTS

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality.

In this system following are the functional requirements:

1. Training dataset must be loaded
2. Hoefflin Anytime tree model must be trained from the dataset
3. User enters the URL in URL locator.
4. URL must be detected and display whether it is legitimate or non-fake website.

4.4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:

1. Product Requirements
2. Organizational Requirements
3. User Requirements
4. Basic Operational Requirements

Some of them are as follows:

- **Reusability**

  The same code with limited changes can be used for detecting phishing attacks variants like smishing, vishing, etc.

- **Maintainability**

  The implementation is very basic and includes print statements that makes it easy to debug.

- **Usability**

  The software used is very user friendly and open source. It also runs on any operating system.

- **Scalability**

  The implementation can include detection of vishing, smishing, etc.

4.4.3 PERFORMANCE REQUIREMENT

a. For desired performance, transferred data size, speed of connection, response time, processing speed must be considered.
b. System should work real – time which means there should be an acceptable time delay between request and response.
c. The system should be reliable to use by the user.

4.4.4 ML PACKAGES

1. **NumPy**

   NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation.

2. **Pandas**

   Pandas is a python package designed for fast and flexible data processing, manipulation and analysis. Pandas has a number of fundamental data structures (a data management and storage format). Pandas Data Frame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

3. **Scikit – learn**

   Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

4. **Matplotlib**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. The various plots we can utilize using Pyplot are Line Plot, Histogram, Scatter, 3D Plot, Image, contour, and Polar.

5. **Seaborn**

Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the Pandas library. The graphs created can also be customized easily. As Matplotlib is also used for the same purpose of Data Visualization, Seaborn uses fascinating themes whereas Matplotlib is used only for Basic Graphs.

6. **Flask**

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

7. **PyMySQL**

PyMySQL is a pure-Python MySQL client library, based on PEP 249. Most public APIs are compatible with mysqlclient and MySQLdb. PyMySQL works with MySQL 5.5+ and MariaDB 5.5+.

4.4.5 ML LIBRARIES

1. **Whois**

pywhois is a Python module for retrieving WHOIS information of domains. pywhois works with Python 2.4+ and no external dependencies.

2. **Xgboost**

   XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library.

3. **Favicon**

   Favicons are used in browser tabs, browser history, toolbar apps, bookmarks dropdown, search bar, and search bar recommendations. In all of these, especially in the bookmarks and history tabs, that consist of lists of URLs all looking the same, the favicon makes it faster to find that web-site you're looking for.

4. **Requests**

   Requests library is one of the integral part of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scraping, requests is must to be learned for proceeding further with these technologies. When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response.

5. **Beautiful soup**

   Python library that is used for web scraping purposes to pull the data out of HTML and XML files. It creates a parse tree from page source code that can be used to extract data in a hierarchical and more readable manner.

6. **Google Search**

   If you want to develop a search service utilizing the power of Google search, you can do so using the google module in Python. You can use this to develop a backend service for a desktop application or implement a website search or app search with the python code running on your server.

4.5 ABOUT THE TECHNOLOGY / SOFTWARE

In this section, the tools and methodology to implement text summarization and classification is detailed.

4.5.1 MACHINE LEARNING

Machine learning is a field of artificial intelligence that focuses on the development of algorithms and models capable of learning patterns from data, enabling systems to make predictions or decisions without explicit programming. It encompasses a variety of techniques, from supervised learning, where models learn from labeled data, to unsupervised learning, where patterns are identified without labeled examples. Machine learning algorithms adapt and

improve their performance over time as they encounter more data, making them versatile tools for tasks such as classification, regression, clustering, and pattern recognition.

In the context of phishing website detection, machine learning is a pivotal tool employed through a systematic process. It begins with the compilation of a labeled dataset, encompassing both phishing and legitimate websites. Features, such as URL structure and content analysis, are then extracted from this dataset. Following data preprocessing, an appropriate machine learning algorithm is selected, and the model is trained to recognize patterns distinguishing between malicious and genuine websites. Validation and hyperparameter tuning ensure the model's efficacy, with evaluation metrics like accuracy and precision guiding the optimization process. Once validated, the model is deployed for real-time detection, often integrated into web browsers or email clients. Continuous monitoring and updates are crucial, given the evolving nature of phishing techniques, and measures are taken to enhance the model's robustness against adversarial attacks. The integration of the machine learning model into broader cybersecurity systems provides a multi-layered defense against phishing threats. This comprehensive approach, combining machine learning with other security measures, strengthens the overall security posture and reduces the risk of falling victim to phishing attacks.

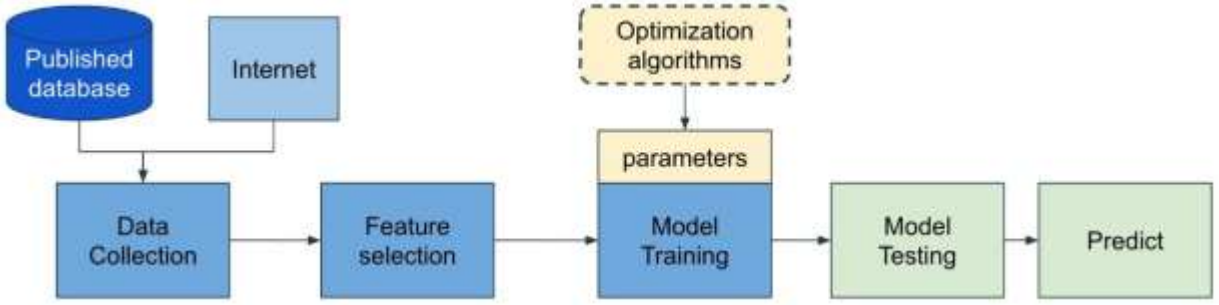

Fig 4.5.1 Phishing Website Detection using Machine Learning

4.5.2 INTRODUCTION TO PYTHON

Python is a powerful high-level, object-oriented programming language created by Guido van Rossum. Python is a general-purpose language. It has a wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

In the late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that

was extensible. This led to the design of a new language which was later named Python.

Python is a great and friendly language to use and learn. and can be adapted to both small and large projects. Python will cut a project's development time greatly and overall, it's much faster to write Python than other languages. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks. Its standard library is made up of many functions that come with Python when it is installed. Represent your data, a few of the most important machine learning algorithms, and how to evaluate the performance of your machine learning algorithm.

4.5.3 PYTHON PROGRAM USING ANACONDA

Use Anaconda Navigator to launch an application. Then, create and run a simple Python program with Jupyter Notebook.

Anaconda is a free and open-source distribution of the Python and R programming language for scientific computing (data science , machine learning applications, large-scale data processing, predictive analytics etc.,)that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS

4.5.3.1 OPEN ANACONDA NAVIGATOR

Choose the instructions for your operating system. Click the Start icon and search for the Navigator

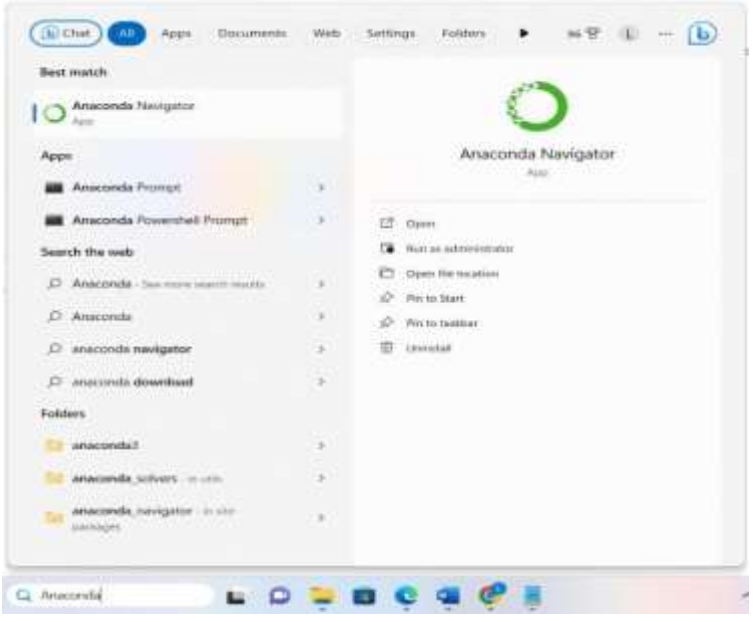

Fig 4.5.3.1 (a) Anaconda Navigator Search

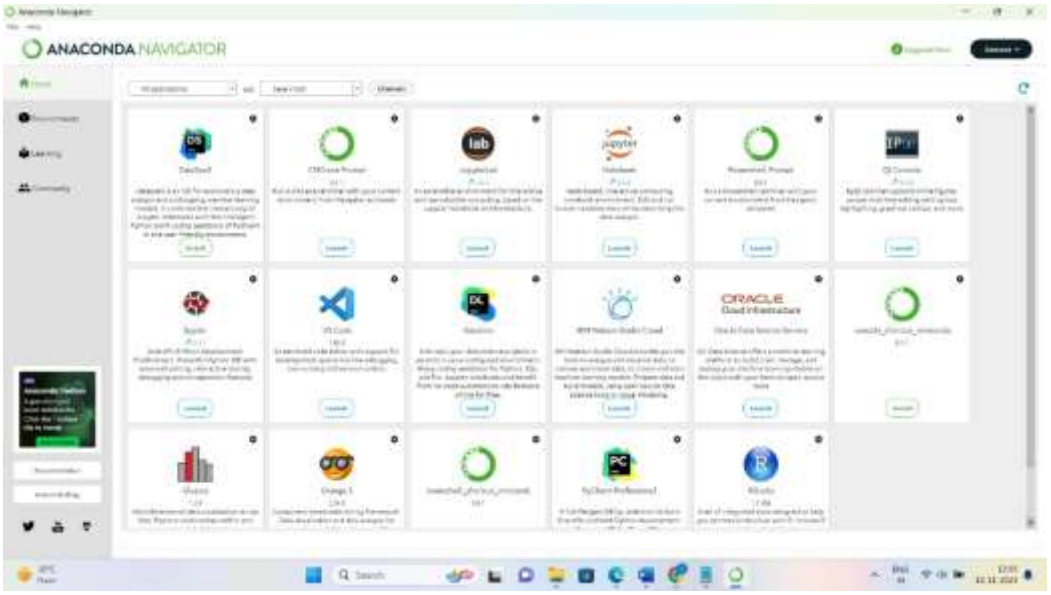

Fig 4.5.3.1 (b) Aanaconda Navigator Homescreen

4.5.3.2 RUN PYTHON IN A JUPYTER NOTEBOOK

On Navigator's Home tab, in the Applications pane on the right, scroll to the Jupyter Notebook tile and click the Install button to install Jupyter Notebook.

NOTE: If Jupyter Notebook is already installed, jump right to the Launch step.

- Launch Jupyter Notebook by clicking Jupyter Notebook's Launch button.
- This will launch a new browser window (or a new tab) showing the Notebook Dashboard.
- On the top of the right hand side, there is a drop down labeled "New".
- Create a new Notebook with the Python version installed.
- In the first line of the Notebook, type or copy/paste print("Hello Anaconda").
- Save Notebook by either clicking the save and checkpoint icon or select File - Save and Checkpoint in the top menu.
- Run new program by clicking the Run button or selecting Cell - Run All from the top menu.

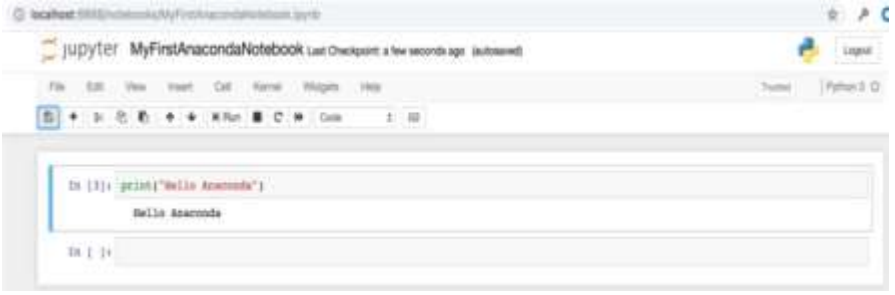

Fig 4.5.3.2 Jupyter Notebook display

4.5.3.3 CLOSE JUPYTER NOTEBOOK

From Jupyter Notebooks' top menu bar, select File - Close and Halt. Then click the Quit button at the upper right of the Notebook Dashboard. Close the window or tab.

4.5.4 PYTHON PROGRAMMING

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management.

It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and C Python are managed by the non-profit Python Software foundation History Guido van Rossum at OSCON 2006.

4.5.5 HISTORY OF PYTHON

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling interfacing with the Amoeba operating system. Its implementation began in December 1989.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were back ported to Python 2.6.x and 2.7.x version series.

Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3. Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.In January 2017, Google announced work on a Python 2.7 to Go trans compiler to improve performance under concurrent workloads. Features and philosophy is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including meta programming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract

and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management.

It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map() and reduce() functions; list comprehensions, dictionaries, sets and generator expressions.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

SYSTEM DESIGN

SYSTEM DESIGN

5.1 MODULE DESCRIPTION

Entire project is divided into 3 modules as follows:

- Data Gathering and pre processing
- Training the model using following Machine Learning algorithms
- Final Prediction Model integrated with frontend

Module 1: Data Gathering and Data Pre processing

a. A proper dataset is searched among various available ones and finalized with the dataset.
b. The dataset must be preprocessed to train the model.
c. In the preprocessing phase, the dataset is cleaned and any redundant values, noisy data and null values are removed.
d. The Preprocessed data is provided as input to the module.

Module 2: Training the model

a. The Preprocessed data is split into training and testing datasets in the 80:20 ratio to avoid the problems of over-fitting and under-fitting.
b. A model is trained using the training dataset with the following algorithms
   i. Logistic Regression
   ii. Random Forest Classifier
   iii. Support Vector Machine Classifier
   iv. K Nearest Neighbor Classifier
   v. Naïve Bayes Classifier
   vi. Multi-Layer Perceptron
   vii. Cat Boost Classifier
   viii. XG Boost Classifier
   ix. Gradient Boosting Classifier
c. The trained models are trained with the testing data and results are visualized using bar graphs, scatter plots.
d. The accuracy rates of each algorithm are calculated using different params like F1 score, Precision, Recall. The results are then displayed using various data visualization tools for analysis purpose.
e. The algorithm which has provided the better accuracy rate compared to remaining algorithms is taken as final prediction model.

```
#from sklearn.externals import joblib
import joblib

# Save the model as a pickle in a file
joblib.dump(model2, 'phishing.pkl')

# Load the model from the file
final_model = joblib.load('phishing.pkl')

pred=final_model.predict(X_test)
```

Fig 5.1: Dumping final prediction ML model

Module 3: Final Prediction model integrated with front end

a. The algorithm which has provided better accuracy rate has considered as the final prediction model.
b. The model thus made is integrated with front end.
c. Database is connected to the front end to store the user information who are using it.
d. The output is printed on the front end as follows:
    i. If the result is -1 the output is given as **"Websie is not safe to use"**.
    ii. Otherwise, the output is displayed as **"Website is safe to use".**

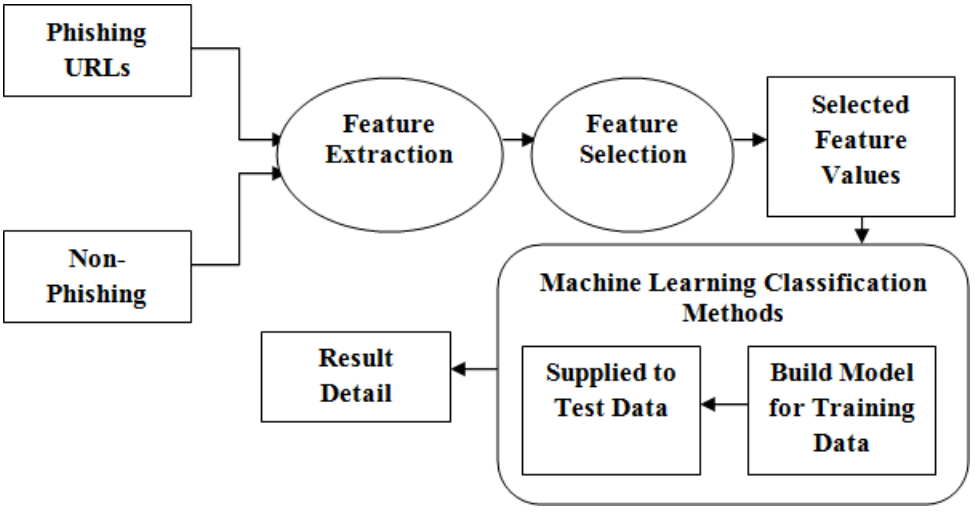

Figure 5.1 Project Flow

5.2 SYSTEM ARCHITECTURE

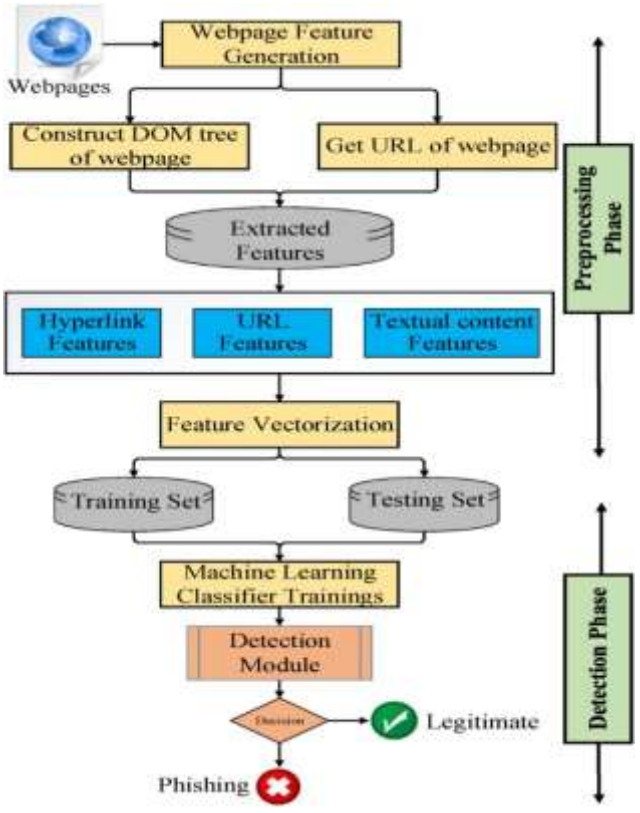

Fig 5.2 System Architecture

Data Collection and URL Retrieval

The system starts by retrieving URLs to be checked for phishing. These URLs can be collected from user input in the webpage created.

URL Validation

The system first validates the user-provided URL to ensure it is well-formed and syntactically correct. This step helps prevent errors due to invalid input.

Feature Extraction

Once the URLs are obtained, the system extracts relevant features from the web pages. These features are essential for training and evaluating the machine learning models. Various features were extracted from the URL database based on Domain, HTML and Address bar of the URLs

Data Preprocessing

The extracted features often need to be preprocessed to ensure consistency and compatibility with the machine learning models.

Machine Learning Model

A machine learning model, trained on a labeled dataset containing both legitimate and phishing URLs, is used to make predictions.

Prediction and Scoring

The machine learning model predicts whether the URL is a phishing site or not. It provides a probability score or a binary classification (phishing or not phishing) based on the trained model's decision boundary.

Result Presentation

The system categorize URLs into "phishing" or "legitimate" and the result is finally displayed on the webpage.

5.3 DATA FLOW DIAGRAMS

DFDs are used to depict graphically the data flow in a system. It explains the processes involved in a system from the input to the report generation. It shows all possible paths from one entity to another of aa system. The detail of a data flow diagram can be represented in three different levels that are numbered 0, 1 and 2.

There are many types of notations to draw a data flow diagram among which Yourdon-Coad and Gane-Sarson method are popular. The DFDs depicted in this chapter uses the Gane-Sarson DFD notations.

5.3.1 DATA FLOW DIAGRAM LEVEL 0

DFD level 0 is called a Context Diagram. It is a simple overview of the whole system being modeled.

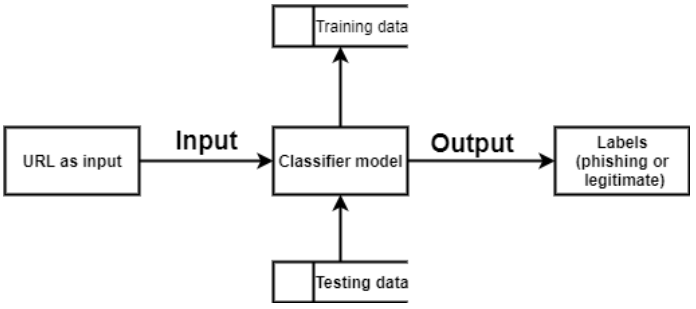

Fig 5.3.1 DFD Level 0

5.3.2 DATA FLOW DIAGRAM LEVEL 1

DFD level 1 gives a more detailed explanation of the Context diagram. The high-level process of the Context diagram is broken down into its subprocesses. The Level 1 DFD takes a step deep by including the processes involved in the system such as feature extraction, splitting of dataset, building the classifier, etc.

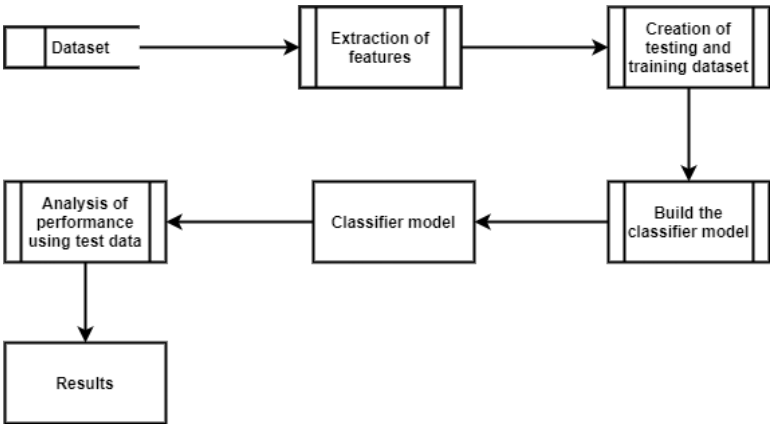

Fig 5.3.2 DFD Level 1

5.3.3 DATA FLOW DIAGRAM LEVEL 2

DFD level 2 goes one more step deeper into the subprocesses of Level 1. Fig 4.4 shows the DFD level 2 of the system. It might require more text to get into the necessary level of detail about the functioning of the system. The Level 2 gives a more detailed sight of the system by categorizing the processes involved in the system to three categories namely preprocessing, feature scaling and classification. It also graphically depicts each of these categories in detail and gives a complete idea of how the system works.

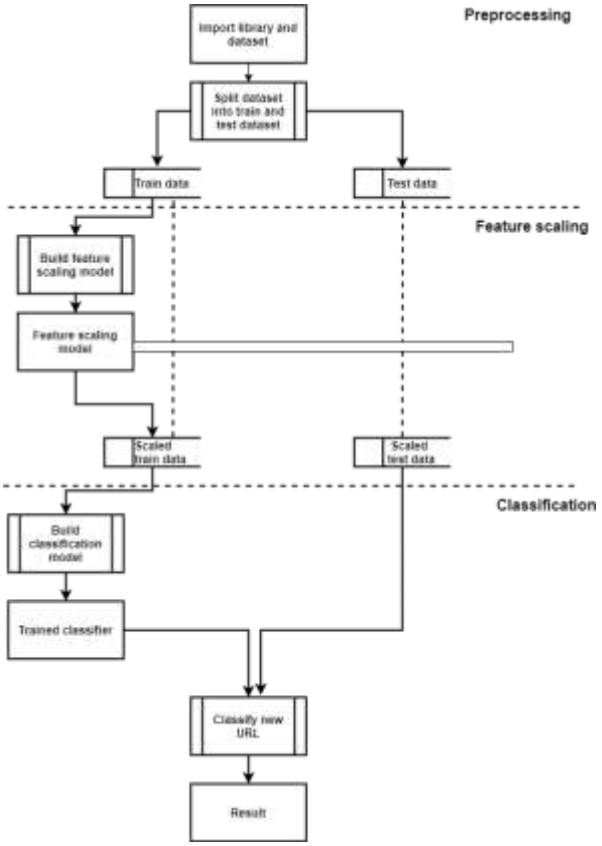

Fig 5.3.3 DFD Level 0

5.4 UML DIAGRAMS

5.4.1 USE CASE DIAGRAM

In UML, use-case diagrams model the behaviour of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use- case diagrams in the early phases of a project and refer to them throughout the development process.

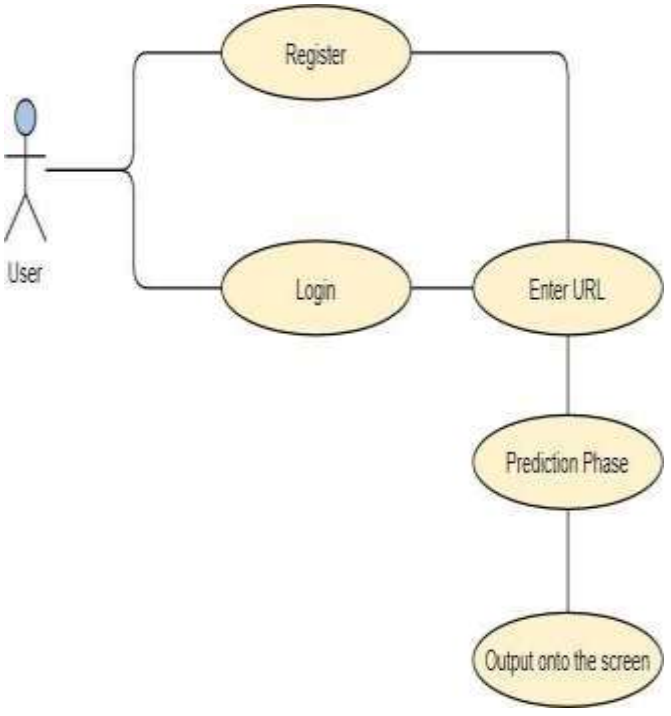

Figure 5.4.1 Use Case Diagram

Figure 5.4.1 depicts how the user interacts with the system to achieve his main objective. User has to register if he's new to the system or login, then he/she has to enter URL they want to check. The URL thus provided is subjected to extraction for features and the extracted features are provided as input to the prediction phase. In the prediction phase, with the saved model (Random Forest Model) the features are checked and the result is printed on the screen. If the result is -1 the output is given as **"Website is not safe to use"**. Otherwise, the output is displayed as **"Website is safe to use".**

5.4.2 ACTIVITY DIAGRAM

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagram are it captures the dynamic behavior of the system. Activity diagram is used to show message flow from one activity to another Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

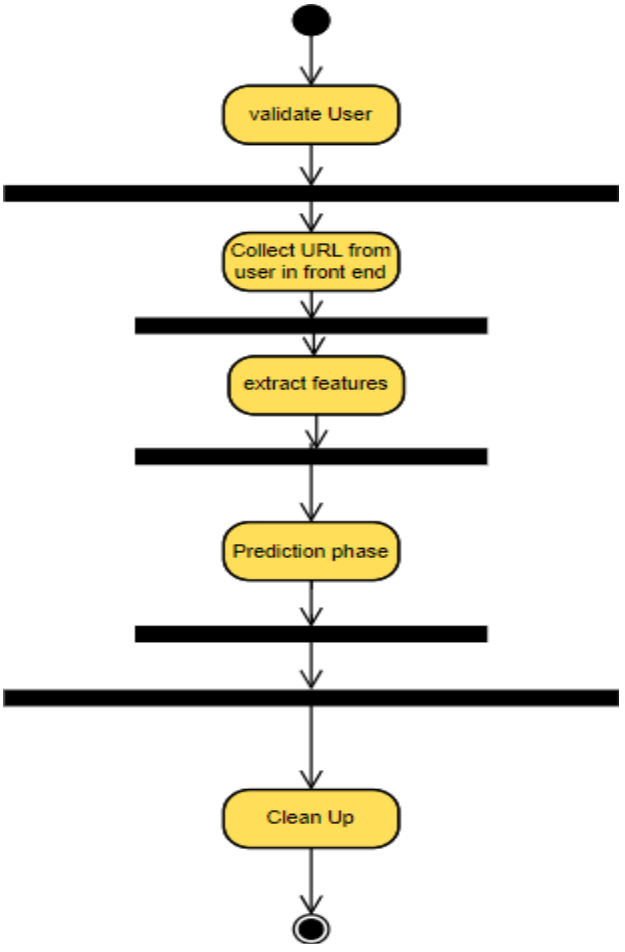

Figure 5.4.2 Activity Diagram

5.4.3 SEQUENCE DIAGRAM

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

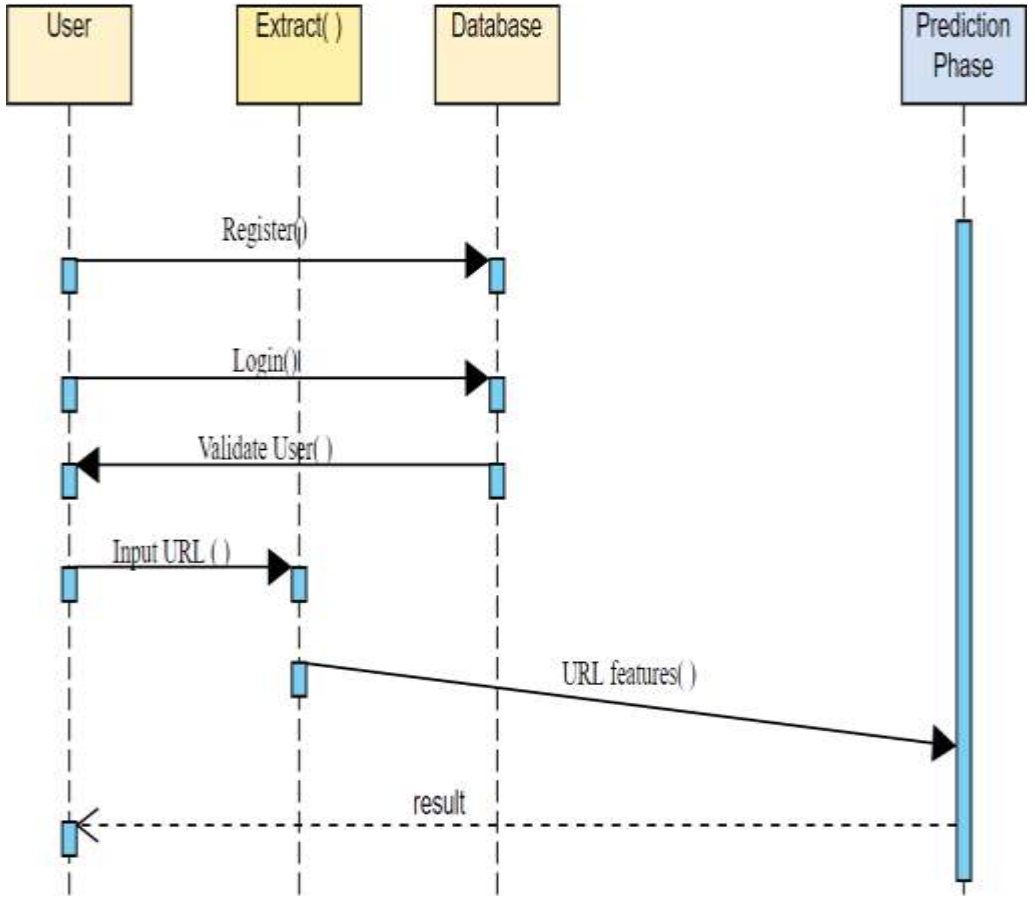

Figure 5.4.3 Sequence Diagram

Figure 5.4.3 depicts the flow of actions by the user to get the final output. User has to first register. Next step is login. When the user provides login credentials, they are checked in the database and validated. If the details are found, the user is moved to the next screen where the URL has to be entered. Else, the user is asked to reenter the credentials. The URL entered is provided as input to extraction phase where the required features are extracted from the URL into a space matrix and this matrix is provided as input to the prediction phase. The predicted result by the ML model is provided as output to the user. If the result is -1 the output is given as **"Website is not safe to use"**. Otherwise, the output is displayed as **"Website is safe to use".**

IMPLEMENTATION AND RESULT ANALYSIS

6. IMPLEMENTATION AND RESULT ANALYSIS

6.1 INTRODUCTION

In the context of phishing website detection, machine learning is a pivotal tool employed through a systematic process. It begins with the compilation of a labeled dataset, encompassing both phishing and legitimate websites. Features, such as URL structure and content analysis, are then extracted from this dataset. Following data preprocessing, an appropriate machine learning algorithm is selected, and the model is trained to recognize patterns distinguishing between malicious and genuine websites. Validation and hyperparameter tuning ensure the model's efficacy, with evaluation metrics like accuracy and precision guiding the optimization process. Once validated, the model is deployed for real-time detection, often integrated into web browsers or email clients. Continuous monitoring and updates are crucial, given the evolving nature of phishing techniques, and measures are taken to enhance the model's robustness against adversarial attacks. The integration of the machine learning model into broader cybersecurity systems provides a multi-layered defense against phishing threats. This comprehensive approach, combining machine learning with other security measures, strengthens the overall security posture and reduces the risk of falling victim to phishing attacks.

6.2 TYPES OF CLASSIFIERS

6.2.1 LOGISTIC REGRESSION

Logistic Regression, despite its name, is a versatile algorithm not limited to binary classification; it can be extended for multi-class text classification tasks. In this context, it works by modeling the relationship between the input features (word frequencies in the case of text) and the probability of a document belonging to each class using the softmax function. The model estimates a separate probability for each class, and the class with the highest probability is assigned as the final prediction.

6.2.2 SUPPORT VECTOR MACHINE

Support Vector Machines (SVMs) are powerful classifiers widely applied to multi-class text classification tasks. SVMs operate by finding an optimal hyperplane in a high-dimensional space that best separates the data points corresponding to different classes. In the context of text classification, each feature represents the frequency of a word in a document, and the SVM seeks to create a decision boundary that maximizes the margin between different classes.

6.2.3 K NEAREST NEIGHBOR CLASSIFIER

K-Nearest Neighbors (KNN) is one of the simplest algorithms used in machine learning for regression and classification problems which is non-parametric and lazy. In KNN there is no need for an assumption for the underlying data distribution. KNN algorithm uses feature similarity to predict the values of new datapoints which means that the new datapoint will be assigned a value based on how closely it matches the points in the training set. The similarity between records can be measured in many different ways. Once the neighbors are discovered, the summary prediction can be made by returning the most common outcome or taking the average. As such, KNN can be used for classification or regression problems. There is no model to speak of other than holding the entire training dataset.

6.2.4 NAIVE BAYES CLASSIFIER

The Naive Bayes classifier is a probabilistic machine learning algorithm based on Bayes' theorem. It assumes that features are independent given the class label, hence the "naive" designation. Widely used in text classification (e.g., spam detection), it calculates the probability of each class given input features and assigns the class with the highest probability as the prediction. Despite its simplicity and the independence assumption, Naive Bayes often performs well in practice and is computationally efficient. It comes in different variants, such as Multinomial, Gaussian, and Bernoulli, suitable for various types of data.

6.2.5 MULTI LAYER PERCEPTRON CLASSIFIER

A Multi-Layer Perceptron (MLP) is an artificial neural network used for supervised learning. It comprises an input layer, hidden layers, and an output layer, with nodes applying activation functions to weighted inputs. During training, the backpropagation algorithm adjusts weights to minimize the difference between predicted and actual outputs. Common activation functions include sigmoid and ReLU. MLPs are versatile, suitable for various tasks like classification and regression, but they require careful tuning to avoid overfitting and efficient optimization algorithms like gradient descent. Their ability to model complex relationships makes them powerful tools in machine learning.

6.2.6 RANDOM FOREST CLASSIFIER

Random Forest, as its name implies, contains a large number of individual decision trees that act as a group to decide the output. Each tree in a random forest specifies the class prediction, and the result will be the most predicted class among the decision of trees. The reason for this amazing result from Random Forest is because of the trees protect each other from individual errors. Although some trees may predict the wrong answer, many other trees will rectify the final prediction, so as a group the trees can move in the right direction.

Random Forests achieve a reduction in overfitting by combining many weak learners that underfit because they only utilize a subset of all training samples Random Forests can handle a large number of variables in a data set. Also, during the forest construction process, they make an unbiased estimate of the generalization error. Besides, they can estimate the lost data well. The main drawback of Random Forests is the lack of reproducibility because the process of forest construction is random. Besides, it is difficult to interpret the final model and subsequent results, because it involves many independent decision trees.

6.2.7 GRADIENT BOOSTING CLASSIFIER

A Gradient Boosting Classifier is a powerful machine learning algorithm that is commonly used for phishing website detection. It belongs to the ensemble learning family and is often employed to improve the accuracy and robustness of binary classification tasks, such as distinguishing between legitimate and phishing websites. Gradient Boosting is an ensemble technique that combines multiple weak learners (typically decision trees) to create a strong predictive model. The primary idea is to iteratively add decision trees, with each new tree correcting the errors made by the previous ones. This process is guided by the gradient of a loss function, which measures the difference between the predicted and actual values

6.2.8 CAT BOOST CLASSIFIER

CatBoost is a powerful and user-friendly algorithm that excels in handling categorical features, making it a valuable tool for classification tasks, especially when dealing with real-world datasets that contain a mix of categorical and numerical features.

6.2.9 XG BOOST CLASSIFIER

XG - Boost is a refined and customized version of a Gradient Boosting to provide better performance and speed. The most important factor behind the success of XG - Boost is its scalability in all scenarios. The XG - Boost runs more than ten times faster than popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The scalability of XG - Boost is due to several important algorithmic optimizations. These innovations include a novel tree learning algorithm for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning. Parallel and distributed computing make learning faster which enables quicker model exploration. More importantly, XG - Boost exploits out of-core computation and enables data scientists to process hundreds of millions of examples on a desktop. Finally, it is even more exciting to combine these techniques to make an end-to-end system that scales to even larger data with the least amount of cluster resources.

6.3 CODE FOR IMPLEMENTATION OF ALGORITHMS

Code For Importing Libraries

```
from sklearn.model_selection import
train_test_split  # Machine Learning Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from catboost import CatBoostClassifier
from sklearn.neural_network import MLPClassifier
```

Code For Training Dataset

```
model1= LogisticRegression()
model2=RandomForestClassifier(random_state = 42, max_depth = 1 5, n_estimators =
200, min_samples_split = 2, min_samples_leaf = 1)
model3=XGBClassifier(n_estimators=500)
model4= KNeighborsClassifier(n_neighbors=7)
model5=DecisionTreeClassifier()
model6= CatBoostClassifier(learning_rate = 0.1)
model7= SVC(kernel = 'linear',gamma = 'scale')
model8=MLPClassifier()
model9= GradientBoostingClassifier(max_depth=n,learning_rate = 0.7)
model1.fit(X_train,  y_train)
model2.fit(X_train, y_train)
model3.fit(X_train, y_train)
model4.fit(X_train, y_train)
model5.fit(X_train, y_train)
model6.fit(X_train, y_train)
model7.fit(X_train, y_train)
```

```
model8.fit(X_train, y_train)
model9.fit(X_train, y_train)
```

Code For Testing Model

```
y_pred1=model1.predict(X_test)
y_pred2 = model2.predict(X_test)
y_pred3 = model3.predict(X_test)
y_pred4 = model4.predict(X_test)
y_pred5 = model5.predict(X_test)
y_pred6 = model6.predict(X_test)
y_pred7 = model7.predict(X_test)
y_pred8 = model8.predict(X_test)
y_pred9 =model9.predict(X_test)
```

Code For Computing The Model Performance

```
acc_train = metrics.accuracy_score(y_train,y_trainmodel)
acc_test = metrics.accuracy_score(y_test,y_testmodel)
print("Accuracy on training Data: {:.3f}".format(acc_train))
print("Accuracy on test Data: {:.3f}".format(acc_test))
f1_score_train = metrics.f1_score(y_train,y_trainmodel)
f1_score_test = metrics.f1_score(y_test,y_testmodel)
print("F1_score on training Data: {:.3f}".format(f1_score_train_svc))
print("F1_score on test Data: {:.3f}".format(f1_score_test_svc))
recall_score_train = metrics.recall_score(y_train,y_trainmodel)
recall_score_test = metrics.recall_score(y_test,y_testmodel)
print("Recall on training Data: {:.3f}".format(recall_score_train))
print("Recall on test Data: {:.3f}".format(recall_score_test))
precision_score_train = metrics.precision_score(y_train,y_trainmodel)
precision_score_test = metrics.precision_score(y_test,y_testmodel)
print("Precision on training Data: {:.3f}".format(precision_score_train))
print("Precision test Data: {:.3f}".format(precision_score_test))
```

Code For Dumping Final ML Model

```
from sklearn.externals import joblib
joblib.dump(model2, 'phishing.pkl')
final_model = joblib.load('phishing.pkl')
pred=final_model.predict(X_test)
```

Code For Displaying Results

```
if  prediction == -1:
    return render_template('index1.html',prediction_text='Website is not safe to use')
else:
    return render_template('index1.html',prediction_text='Website is not safe to use')
```

6.4 COMPARISION OF MODELS

When comparing different machine learning classifiers, it is crucial to assess their performance using a variety of metrics to gain a comprehensive understanding of their effectiveness. Four key metrics for evaluation are accuracy, F1 score, precision, and recall. Accuracy, representing the ratio of correctly predicted instances to the total instances, is suitable for balanced datasets but may not be appropriate for imbalanced ones. The F1 score, being the harmonic mean of precision and recall, is valuable in cases of uneven class distribution. Precision, the ratio of correctly predicted positive observations to the total predicted positives, is important when false positives are costly. Recall, or the true positive rate, is crucial when false negatives are undesirable. The choice between precision and recall depends on the specific requirements of the application; for instance, medical diagnoses may prioritize minimizing false negatives. It's essential to consider the trade-off between precision and recall and to be mindful of the context of the problem being addressed. Evaluating classifiers should involve a combination of these metrics, especially in scenarios with imbalanced datasets, where accuracy alone may be misleading. Employing techniques like cross-validation enhances the robustness of performance evaluations.

After organizing the performance metrics for different machine learning classifiers into a Data Frame, the process of selecting the best model involves several key steps. Firstly, it's important to calculate aggregated measures such as the mean for each metric across multiple runs or folds of cross-validation. This helps mitigate the impact of randomness in the training process. Subsequently, the Data Frame can be sorted based on the aggregated metric values, and visual inspection through plots or graphs can provide an intuitive understanding of how

models compare. The criteria for selecting the best model should be defined, taking into account the specific nature of the problem. This might involve prioritizing a particular metric, depending on whether accuracy, precision, or recall is of greater importance. Cross-validation is a critical step to ensure the robustness of performance metrics, as it provides estimates of model performance on different subsets of the data. Ensemble methods, such as bagging or boosting, can be explored to combine predictions from multiple models and potentially improve overall performance. Additionally, domain knowledge and practical considerations should play a role in the decision-making process. Sometimes, a model that slightly underperforms according to traditional metrics may be more suitable for deployment based on other factors like interpretability or resource requirements. Hyperparameter tuning can be performed on the selected model(s) to further optimize performance, and the final step involves evaluating the chosen model(s) on a separate test set not used during training or model selection. This step provides an unbiased estimate of the model's generalization performance and ensures that the selected model performs well on new, unseen data. The entire process of model selection is iterative, and it's important to continuously refine and validate choices based on ongoing analyses and feedback from the real-world application.

Table 6.4: Comparison of models

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	XGBoost Classifier	0.969	0.973	0.993	0.984
3	Multi-layer Perceptron	0.969	0.973	0.995	0.981
4	Random Forest	0.967	0.971	0.993	0.990
5	Support Vector Machine	0.964	0.968	0.980	0.965
6	Decision Tree	0.960	0.964	0.991	0.993
7	K-Nearest Neighbors	0.956	0.961	0.991	0.989
8	Logistic Regression	0.934	0.941	0.943	0.927
9	Naive Bayes Classifier	0.605	0.454	0.292	0.997

Hence Gradient Boosting Classifier Algorithm is considered as the best model among them and it is selected as the final model. Now to save this trained model in Python, we can use the “pickle” or “joblib” module.

6.5 WHY GRAIDIENT BOOSTING ALGORITHM

A Gradient Boosting Classifier is a powerful machine learning algorithm that is commonly used for phishing website detection. It belongs to the ensemble learning family and is often employed to improve the accuracy and robustness of binary classification tasks, such as distinguishing between legitimate and phishing websites. Gradient Boosting is an ensemble technique that combines multiple weak learners (typically decision trees) to create a strong predictive model. The primary idea is to iteratively add decision trees, with each new tree correcting the errors made by the previous ones. This process is guided by the gradient of a loss function, which measures the difference between the predicted and actual values.

Gradient Boosting is known for its high predictive accuracy, which is crucial for the accurate detection of phishing websites. It leverages the strength of an ensemble of decision trees and iterative error correction to achieve high accuracy and robustness in classifying websites as either legitimate or malicious.

6.6 DEPLOYMENT

To deploy a machine learning model for phishing website detection, the initial steps involve saving the trained model through serialization methods like pickle or joblib. Following this, a web application or API needs to be developed to serve as the user interface for interacting with the model. Once the model is saved, it is loaded into the deployment environment's memory to facilitate access for making predictions. Input data handling mechanisms are then implemented to collect relevant information for the phishing detection task, such as URL characteristics or user behavior. Ensuring consistency, the same preprocessing steps applied during model training are then used on incoming data. The loaded model is employed for inference, providing predictions on whether a given website is classified as phishing or legitimate. Finally, if deploying a web application, integration with the frontend is crucial for a seamless user experience, while APIs should be designed for easy integration into other systems. These foundational steps collectively establish the deployment environment, enabling the model to effectively process incoming data and deliver accurate predictions in a real-world setting.

Python Flask Code

Main.py

```
flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics import warnings
import pickle
from convert import convertion warnings.filterwarnings('ignore')
from feature import FeatureExtraction
file = open("newmodel.pkl","rb")
gbc = pickle.load(file) file.close()
app = Flask( name ) @app.route("/")
def home():
    return render_template("index.html")
@app.route('/result',methods=['POST','GET'])
def predict():
    url = request.form["name"]
    obj = FeatureExtraction(url)
    x = np.array(obj.getFeaturesList()).reshape(1,30)
    y_pred =gbc.predict(x)[0]
    name=convertion(url,int(y_pred))
    return render_template("index.html", name=name)
@app.route('/usecases', methods=['GET', 'POST'])
def usecases():
    return render_template('usecases.html')
if  name == " main ":
    app.run(debug=True)
```

Convert.py

```
def convertion(url,prediction):
    name = []
    if(prediction==1):
        return [url,"Safe","Continue"]
    else:
        return [url,"Not Safe","Still want to Continue"]
```

6.7 PREDICTION

In the deployed website for phishing website detection, the prediction process unfolds in several sequential steps. Users initiate the process by providing input data, typically in the form of a URL or relevant features associated with a website. Following this, the input data undergoes preprocessing to ensure proper formatting and alignment with the model's training data. The pre-trained machine learning model, previously saved and deployed, is then loaded into the website's memory, allowing access for making predictions. If necessary, feature extraction or additional processing may occur to obtain relevant information for the model. Subsequently, the machine learning model is invoked to perform inference, applying learned patterns to classify the website as either legitimate or potentially a phishing site. The resulting prediction is communicated back to the user through the website's interface, enabling users to take appropriate actions based on the model's assessment. The predicted result by the ML model is provided as output to the user. If the result is -1 the output is given as **"Website is not safe to use"**. Otherwise, the output is displayed as **"Website is safe to use".** Optionally, the website may incorporate a feedback mechanism, providing users with the opportunity to contribute input on the accuracy of predictions, thereby facilitating continuous improvement and retraining of the machine learning model. This user-friendly and efficient process allows for quick assessments of a website's legitimacy based on the insights generated by the deployed machine learning model.

TESTING

7. TESTING

7.1 METHODS OF TESTING

The main aim of the testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operating environments.

In this project, we have developed a GUI and a Machine Learning code which helps in detecting Website URL's and predicting them phished or not. The main aim of testing this project is to check if the URL is being predicted accurately and check the working performance when different URLs are given as inputs.

The testing steps are:

a. Unit Testing
b. Integration Testing
c. Validation Testing
d. User Acceptance Testing
e. Output testing

7.1.1 UNIT TESTING

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time and costs. The following Unit testing table shows the functions that were tested at the time of programming. The first column gives all the modules which were tested, and the second column gives the test results. Test results indicate if the functions, for given inputs are delivering valid outputs.

7.1.2 VALIDATION TESTING

At the culmination of integration testing, software is completed assembled as a package. Interfacing errors have been uncovered and corrected. Validation testing can be defined in many ways; here the testing validates the software function in a manner that is reasonably expected by the customer. In software project management, software testing, and software

engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control.

7.1.3 FUNCTIONAL TESTING

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

Table 7.1: Functional testing

Function Name	**Tests Results**	**Expected Output**	**Actual Output**
Uploading Code	Tested for accuracy with different algorithms	Output of Accuracy rate of Algorithms used	Output of Accuracy rate of Algorithms used
Get Accuracy rate	Calculating Accuracy rate with ML model	Analysis Graph	Analysis Graph
Display Result	Output is displayed successfully for the given websites	Phishing detection is done according to the ML model corrected and output is printed on Screen	Phishing detection is done according to the ML model corrected and output is printed on Screen

7.1.4 INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together. Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

7.1.5 USER ACCEPTANCE TESTING

Performance of an acceptance test is actually the user's show. User motivation and knowledge are critical for the successful performance of the system. The above tests were conducted on the newly designed system performed to the expectations. All the above testing strategies were done using the following test case designs.

7.2 TEST CASES

Table 7.1: Parameters Extraction Test

Test Case	1
Name of Test	Extracting Parameters
Input	https://www.google.com
Expected Output	Parameters are extracted and values are assigned
Actual Output	Parameters are extracted and values are assigned
Result	Successful

Table 7.2: Prediction Test 1

Test Case	1
Name of Test	Prediction Test
Input	https://www.google.com
Expected Output	Website is safe to use
Actual Output	Website is safe to use
Result	Successful

Table 7.3: Prediction Test 2

Test Case	2
Name of Test	Prediction Test
Input	https://Ieeexplore.ieee.org
Expected Output	Website is safe to use
Actual Output	Website is safe to use
Result	Successful

Table 7.4: Prediction Test 3

Test Case	3
Name of Test	Prediction Test
Input	http://123.456.789.123/amazon.com/
Expected Output	Website is not safe to use
Actual Output	Website is not safe to use
Result	Successful

Table 7.5: Prediction Test 4

Test Case	4
Name of Test	Prediction Test
Input	http://123.456.789.123/paypal.com/
Expected Output	Website is not safe to use
Actual Output	Website is not safe to use
Result	Successful

7.3 SCREENSHOTS

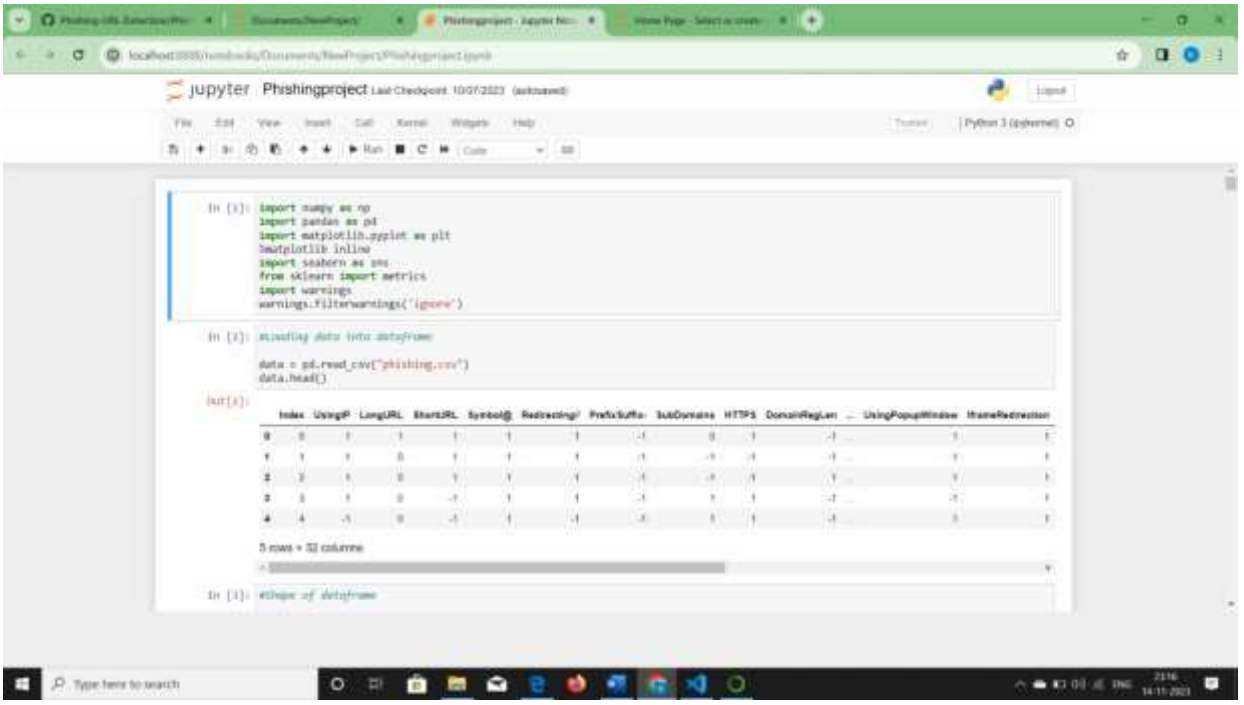

Fig 7.3(a) Jupyter Notebook

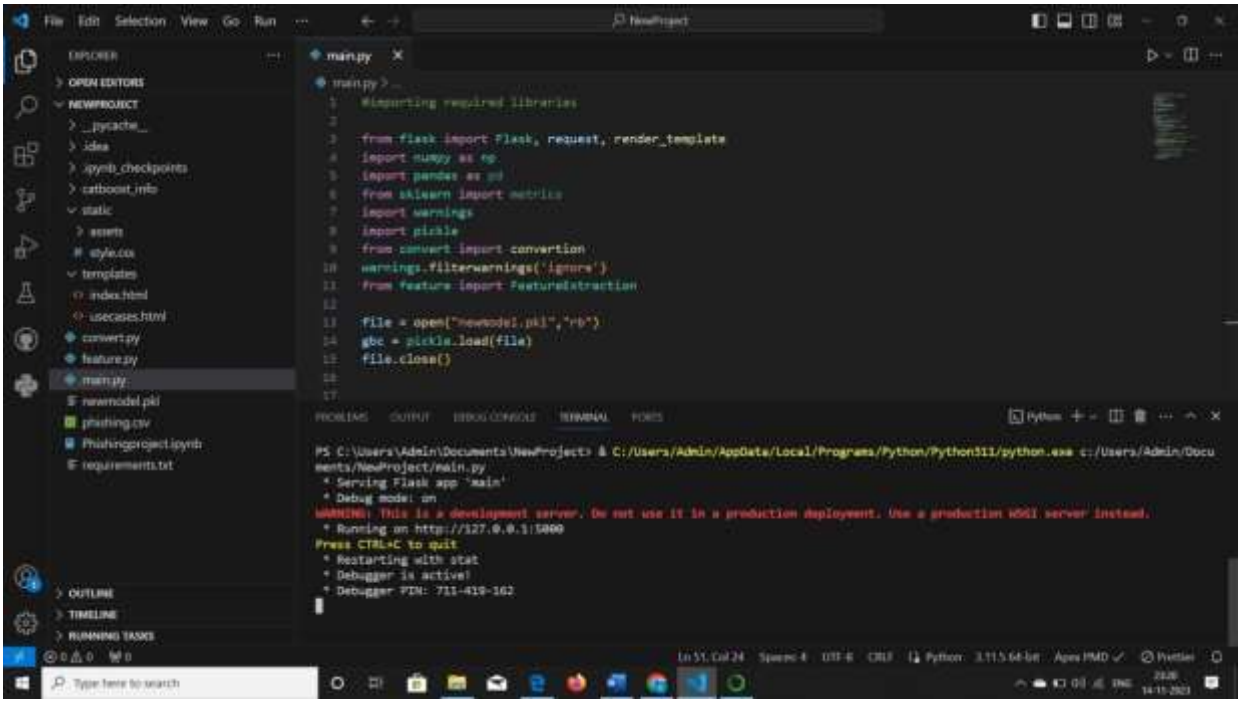

Fig 7.3(b) Snapshot of link to the website

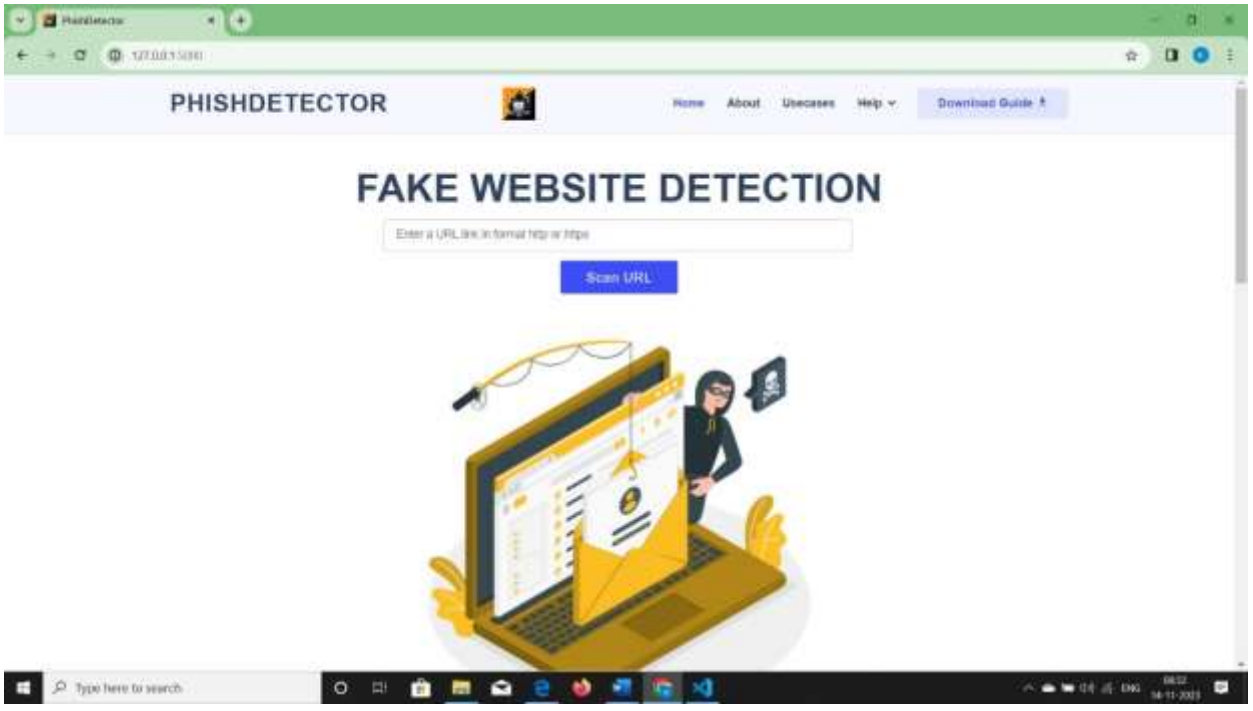

Fig 7.3(c) URL Enter page

Fig 7.3(d) About page

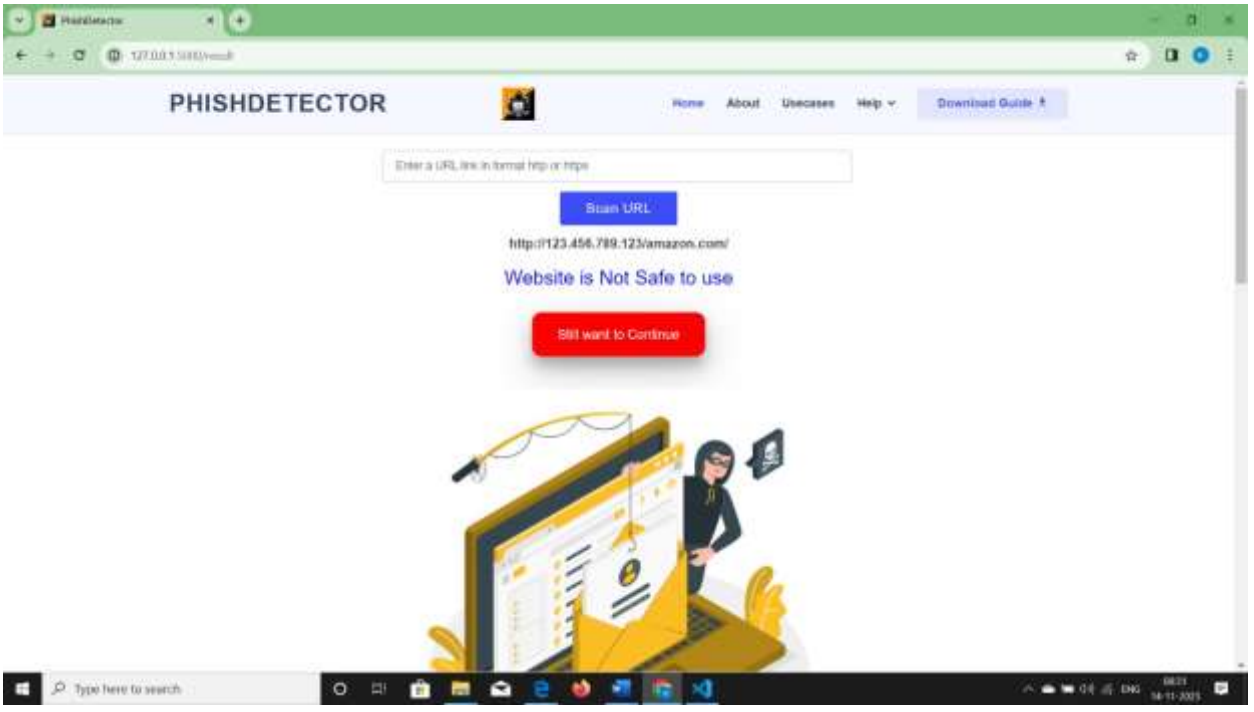

Fig 7.3(e) Output snapshot 1

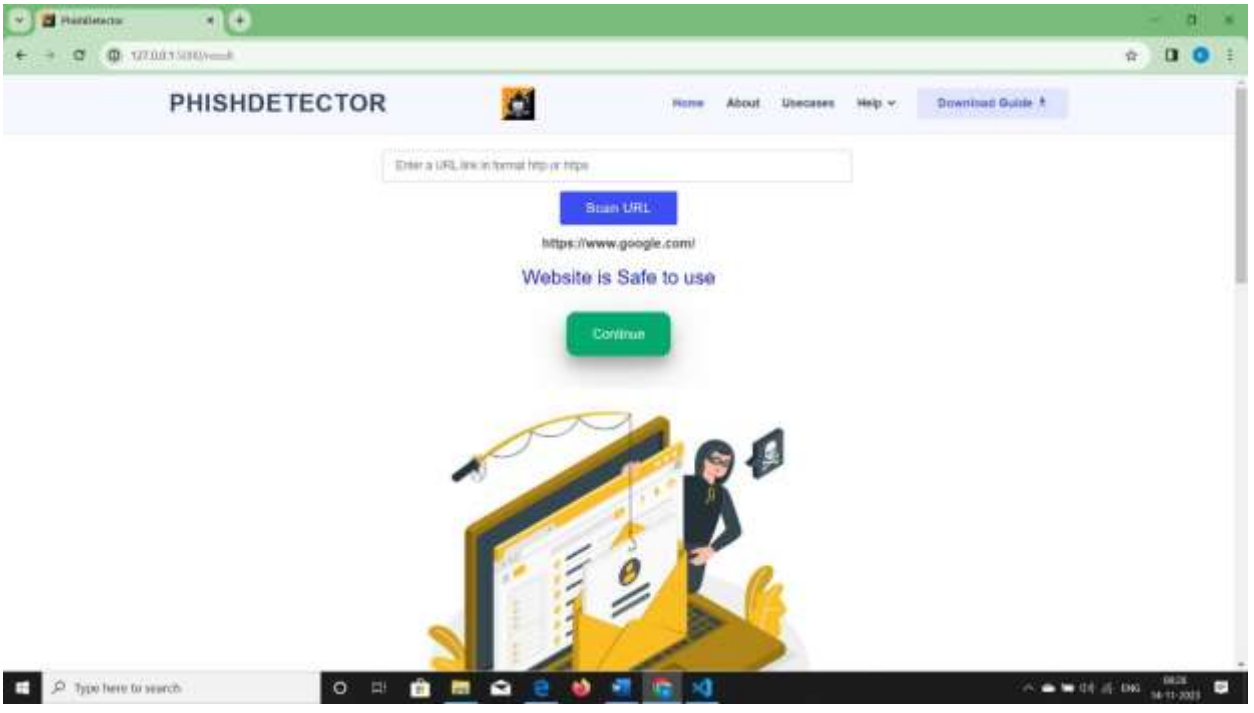

Fig 7.3(f) Output snapshot 2

CONCLUSION AND FUTURE ENHANCEMENT

8. CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

It is found that phishing attacks is very crucial and it is important for us to get a mechanism to detect it. As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using any of the machine learning algorithm with the classifier. We already have classifiers which gives good prediction rate of the phishing besides, but after our survey that it will be better to use a hybrid approach for the prediction and further improve the accuracy prediction rate of phishing websites. We have seen that existing system gives less accuracy so we proposed a new phishing method that employs URL based features and also, we generated classifiers through several machine learning. We have got the desired results of testing the site is phishing or not by using five different classifiers.

8.2 FUTURE ENHANCEMENT

Further work can be done to enhance the model by using ensembling models to get greater accuracy score. Ensemble methods is a ML technique that combines many base models to generate an optimal predictive model. Further reaching future work would be combining multiple classifiers, trained on different aspects of the same training set, into a single classifier that may provide a more robust prediction than any of the single classifiers on their own. The project can also include other variants of phishing like smishing, vishing, etc. to complete the system. Looking even further out, the methodology needs to be evaluated on how it might handle collection growth. The collections will ideally grow incrementally over time so there will need to be a way to apply a classifier incrementally to the new data, but also potentially have this classifier receive feedback that might modify it over time.

REFERENCES

[1] J. Rashid, T. Mahmood, M. W. Nisar and T. Nazir, "Phishing Detection Using Machine Learning Technique," 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), 2020, pp. 43-46.

[2] M. H. Alkawaz, S. J. Steven and A. I. Hajamydeen, "Detecting Phishing Website Using Machine Learning," 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 2020, pp. 111-114.

[3] V. Patil, P. Thakkar, C. Shah, T. Bhat and S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5.

[4] W. Bai, "Phishing Website Detection Based on Machine Learning Algorithm," 2020 International Conference on Computing and Data Science (CDS), 2020, pp. 293-298.

[5] A. Razaque, M. B. H. Frej, D. Sabyrov, A. Shaikhyn, F. Amsaad and A. Oun, "Detection of Phishing Websites using Machine Learning," 2020 IEEE Cloud Summit, 2020, pp. 103-107.

[6] M. M. Vilas, K. P. Ghansham, S. P. Jaypralash and P. Shila, "Detection of Phishing Website Using Machine Learning Approach," 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT), 2019, pp. 384-389.

[7] A. Alswailem, B. Alabdullah, N. Alrumayh and A. Alsedrani, "Detecting Phishing Websites Using Machine Learning," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), 2019, pp. 1-6.

[8] Yuan, H., Chen, X., Li, Y., Yang, Z., & Liu, "Detecting Phishing Websites and Targets Based on URLs and Webpage Links," 2018 24th International Conference on Pattern Recognition , 2018, pp. 3669-3674.

[9] SHENG, Steve; WARDMAN, Brad; WARNER, Gary; CRANOR, Lorrie; HONG, Jason; ZHANG, Chengshan. An Empirical Analysis of Phishing Blacklists. 2009.

[10] Phishing Activity Trends Report Q4 2018 [online]. APWG [visited on 2020-04-13]. Available from: https://docs.apwg.org/ /reports/apwg_trends_report_q4_2018.pdf.

[11] GUARNIERI, Claudio. The Year of the Phish [online]. Nex [visited on 2020-04-12]. Available from: https://nex.sx/blog/2019/ 12/15/the-year-of-the-phish.html.

[12] Phishing Activity Trends Report [online].

[13] Uniform Resource Identifier (URI): Generic Syntax [online]. IETF Available from: https://tools.ietf.org/html/rfc3986

[14] KOZA, John R.; BENNETT, Forrest H.; ANDRE, David; KEANE, Martin A. Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. In: Artificial Intelligence in Design '96. 1996, pp. 151–170. ISBN 978-94-009-0279-4. Available also from: https://doi.org/10.1007/978- 94-009-0279-4_9.

[15] R Kiruthiga and D. Akila ,"Phishing Website Detection Using Machine Learning", 2022.

[16] GERÓN, Aurelién. Hands-On Machine Learning with ScikitLearn, Keras, and TensorFlow. In: O'Reilly Media, Inc., 2017, chap. 1. ISBN 978-1-49-203264-9.

[17] Transport Layer Security (TLS) Extensions [online]. IETF [visited on 2020-04-18]. Available from: https://www.rfc-editor.org/ info/rfc3546.

[18] Lizhen Tang and Qusay H. Mahmoud, "A Survey of Machine Learning Based Solutions for Phishing website Detection", 2021

[19] COX, Nicholas; JONES, Kelvyn. Exploratory data analysis. Quantitative Geography, London: Routledge. 1981, pp. 135–143.

[20] HUCKA, Michael. Nostril: A nonsense string evaluator written in Python. Journal of Open Source Software. 2018, vol. 3, no. 25, pp. 596. Available from DOI: 10.21105/joss.00596.

[21] CLAESEN, Marc; DE MOOR, Bart. Hyperparameter Search in Machine Learning. 2015.