

**IBM**  
**NALAIYA THIRAN**

**PROJECT REPORT**  
**ON**  
**WEB PHISHING DETECTION**

**TEAM ID: PNT2022TMID42127**

**AVS COLLEGE OF TECHNOLOGY**

*Team Members*

**NIVETHA P**  
**INDUMATHI C**  
**PRITHIKA K**  
**HEMALATHA V**

## **TABLE OF CONTENTS**

### **1. INTRODUCTION**

1.1 Project Overview

1.2 Purpose

### **2. LITERATURE SURVEY**

2.1 Existing problem

2.2 References

2.3 ProblemStatement Definition

### **3. IDEATION & PROPOSED SOLUTION**

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 ProblemSolution fit

### **4. REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

### **5. PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

### **6. PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

### **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

7.1 Feature 1

7.2 Feature 2

7.3 DatabaseSchema (if Applicable)

### **8. TESTING**

8.1 Test Cases

8.2 User Acceptance Testing

### **9. RESULTS**

9.1 Performance Metrics

### **10. ADVANTAGES & DISADVANTAGES**

### **11. CONCLUSION**

### **12. FUTURE SCOPE**

### **13. APPENDIX**

Source Code

GitHub & Project Demo Link

## **ABSTRACT**

Phishing is the most commonly used social engineering and cyber attack. Through such attacks, the phisher targets naive online users by tricking them into revealing confidential information, with the purpose of using it fraudulently. In order of website being detected as phishing. Detect them in their early appearance, using machine learning and deep neural network algorithms. Of the above three, the machine learning based method is proved to be most effective than the other methods. A phishing website is a common social engineering method that the dataset created to predict phishing websites. Both phishing and benign URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted. The performance level of each model is measured and compared.

**Keywords:** Deep learning, Machine learning, Phishing website attack, Phishing website detection, Anti-phishing website, Legitimate website , Phishing website datasets, Phishing website features.

## **PRE-REQUISITES**

**TOOLS : JUPITER**

**NOTEBOOK OPERATING**

**SYSTEM : WINDOWS 10**

**LANGUAGE : PYTHON**

## **INSTALLING LIBRARIES**

In this first step, we have to import the most common libraries used in python for machine learning such as

- ☐ Pandas
- ☐ Numpy
- ☐ Seaborn
- ☐ Matplotlib
- ☐ Sklearn
- ☐ Flask

## **IMPORTING DATA**

In this project, we have used the url pre processed data.

## **CHAPTER 1**

### **INTRODUCTION**

Phishing imitates the characteristics and alternatives of emails and makes it appear similar due to the fact the original one. It seems nearly like that of the legitimate supply. The consumer thinks that this e-mail has come back from a real employer or a corporation. This makes the consumer to forcefully visit the phishing internet site thru the hyperlinks given inside the phishing email. These phishing web sites region unit created to mock the seams of an ingenious website. The phishers force person to inventory up the non- public info via giving baleful messages or validate account messages etc. so that they inventory up the preferred data which might be utilized by them to misuse it. They devise things such as the user isn't always left with the other choice but to go to their spoofed web site. Phishing is the most hazardous criminal physical activities in the cyber region. Since the maximum of the customers logs on to get admission to the services supplied with the aid of government and financial establishments, there has been a significant boom in phishing attacks for the beyond few years. Phishers commenced to earn cash and that they try this as a thriving business.

Phishing may be law-breaking, the explanation behind the phishers doing this crime is that it is terribly trustworthy to try to do this, it doesn't value something and it effective. The phishing will truly get entry to the e-mailidentity of somebody it's terribly sincere to are looking for out the email identification currently every day and you will send an email to every person is freely

offered throughout the globe. These attacker's vicinity terribly much less price and electricity to urge valuable know-how quick and truly. The phishing frauds effects malware infections, statistics loss, fraud, etc. information at some stage

in which those cyber criminals have an interest is that the crucial data of a user similar to the password, OTP, credit/ debit card numbers CVV, sensitive know- how associated with business, medical understanding, confidential information, etc commonly these criminals conjointly acquire data which may provide them directly get admission to do the social media account their emails.

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.

## **1.1 PROJECT OVERVIEW**

- ☐ To develop a novel approach to detect malicious URL and alert users.
- ☐ To apply ML techniques in the proposed approach in order to analyze therealtime URLs and produce effective results.
- ☐ To implement the concept of RNN, which is a familiar ML technique thathasthecapability to handle huge amount of data.

## **1.2 PURPOSE**

- To develop an unsupervised deep learning method to generate insight from aURL.
- The study can be extended in order to generate an outcome for a largernetwork and protect the privacy of an individual.

## CHAPTER 2

### **LITERATURE SURVEY**

Rathore, S., Sharma, P.K., Loia, V., Jeong, Y.S. and Park, J.H. [11] presents a comprehensive survey of different security and privacy threats that target every user of social networking sites. A Social Network Service (SNS) is a type of web service for establishing a virtual connection between people with similar interests, backgrounds, and activities. In recent years, SNSs become a well-liked medium of communication. The number of SNS users worldwide is continuously increasing every year. This paper separately focuses on various threats that arise due to the sharing of multimedia content within a social networking site. In this, describing three classes of threats – Multimedia Content Threats, Traditional Threats and Social Threats Pujara, P. and Chaudhari, M.B., [10] Phishing frauds might be the most popular cybercrime used today. This paper detailed literature survey and proposed new approach to detect phishing website by features extraction and machine learning algorithm. In this paper author describe different methodologies such as Blacklist method, Heuristic based method, Visual similarity and Machine learning for phishing detection. Blacklist method is used in which list of phishing URL is stored in database and then if URL is found in database, it is known as phishing URL and gives warning otherwise it is called legitimate. Heuristic based method is extension of blacklist and able to detect new attack as use features extracted from phishing site to detect phishing attack. Visual similarity approach deceive user by extracting image of legitimate site. Machine Learning approach works efficiently in large dataset Jain, A.K. and Gupta, B.B., Attackers steal sensitive information like personal identification number (PIN), credit card details, login, password, etc., from internet users. In this paper, author proposed a machine learning based anti-phishing system based on Uniform Resource Locator (URL) features. To evaluate the performance of proposed system, author taken 14 features from URL to detect a website as a phishing or



non- phishing. The proposed system is trained using quite 33,000 phishing and legitimate URLs with SVM and Naïve Bayes classifiers. Experiment results show quite 90% accuracy in detecting phishing websites using SVM classifier

## REFERENCES

- [1] Godbole, N. and Belapure, S., 2011. Cyber Security, Understanding Computer Forensics and Legal Perspectives.
- [2] <https://apwg.org/trendsreports/>.
- [3] <https://computer.howstuffworks.com/phishing.htm>.
- [4] <https://inspiredelearning.com/blog/social-phishing/>
- [5] <https://timesofindia.indiatimes.com/city/vadodra/multi-state-job-racket-busted-by-cybercrime-cell/articleshow/66421586.cms>
- [6] <https://www.webopedia.com/DidYouKnow/Internet/phishing.asp>.
- [7] Jain, A.K. and Gupta, B.B., 2019. A machine learning based approach for phishing detection using hyperlinks information. *Journal of Ambient Intelligence and Humanized Computing*, 10(5), pp.2015-2028.
- [8] Jain, A.K. and Gupta, B.B., 2018. PHISH-SAFE: URL features- based phishing detection system using machine learning. In *Cyber Security* (pp. 467-474). Springer, Singapore.
- [9] Machado, L. and Gadge, J., 2017, August. Phishing Sites Detection Based on C4. 5 Decision Tree Algorithm. In *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (pp. 1-5). IEEE.
- [10] Pujara, P. and Chaudhari, M.B., 2018. Phishing Website Detection using Machine Learning: A Review.
- [11] Rathore, S., Sharma, P.K., Loia, V., Jeong, Y.S. and Park, J.H., 2017. Social network security: Issues, challenges, threats, and solutions. *Information sciences*, 421, pp.43-69.

## **2.1 EXISTING PROBLEM**

In this technological era, the Internet has made its way to become an inevitable part of our lives. It leads to many convenient experiences in our lives regarding communication, entertainment, education, shopping and so on. As we progress into online life, criminals view the Internet as an opportunity to transfer their physical crimes into a virtual environment. The Internet not only provides convenience in various aspects but also has its downsides, for example, the anonymity that the Internet provides to its users. Presently, many types of crimes have been conducted online. Hence, the main focus of our research is phishing. Phishing is a type of cybercrime where the targets are lured or tricked into giving up sensitive information, such as Social Security Number personal identifiable information and passwords. This obtainment of such information is done fraudulently. Given that phishing is a very broad topic, we have decided that this research should specifically focus on phishing websites.

Rao et al. [1] proposed a novel classification approach that use heuristic-based feature extraction approach. In this, they have classified extracted features into three categories such as URL Obfuscation features, Third-Party-based features, Hyperlink-based features. Moreover, proposed technique gives 99.55% accuracy. Drawback of this is that as this model uses third party features, classification of website dependent on speed of third-party services. Also this model is purely depends on the quality and quantity of the training set and Broken links feature extraction has a Volume 3.

Chunlin et al. [2] proposed approach that primarily focus on character frequency

features. In this they have combined statistical analysis of URL with machine learning technique to get result that is more accurate for classification of malicious URLs. Also they have compared six machine-learning algorithms to verify the effectiveness of proposed algorithm which gives 99.7% precision with false positive rate less than 0.4%. Sudhanshu et al. [3] used association data mining approach. They have proposed rule based classification technique for phishing website detection. They have concluded that association classification algorithm is better than any other algorithms because of their simple rule transformation. They achieved 92.67% accuracy by extracting 16 features but this is not up to mark so proposed algorithm can be enhanced for efficient detection rate.

M. Amaad et al.[4] presented a hybrid model for classification of phishing website. In this paper, proposed model carried out in two phase. In phase 1, they individually perform classification techniques, and select the best three models based on high accuracy and other performance criteria. While in phase 2, they further combined each individual model with best three model and makes hybrid model that gives better accuracy than individual model. They achieved 97.75% accuracy on testing dataset. There is limitation of this model that it requires more time to build hybrid model.

Hossein et al.[5] developed an open-source framework known as “Fresh-Phish”. For phishing websites, machine-learning data can be created using this framework. In this, they have used reduced features set and using python for building query. They build a large labelled dataset and analyse several machine-learning classifiers against this dataset. Analysis of this gives very good accuracy using machine-learning classifiers. These analyses how long time it takes to train the model.

Gupta et al. [6] proposed a novel anti phishing approach that extracts features from client-side only. Proposed approach is fast and reliable as it is not dependent on third party but it extracts features only from URL and

source code. In this paper, they have achieved 99.09% of overall detection accuracy for phishing website. This paper have concluded that this approach has limitation as it can detect webpage written in HTML

.Non-HTML webpage cannot detect by this approach.

Bhagyashree et al.[7] proposed a feature based approach to classify URLs as phishing and nonphishing. Various features this approach uses are lexical features, WHOIS features, Page Rank and Alexa rank and Phish Tank-based features for disguising phishing and non-phishing website. In this paper, web-mining classification is used. Mustafa et al.[8] developed safer framework for detecting phishing website. They have extracted URL features of website and using subset based selection technique to obtain better accuracy .In this paper, author evaluated CFS subset based and content based subset selection methods And Machine learning algorithms are used for classification purpose.

Priyanka et al.[9] proposed novel approach by combining two or more algorithms. In this paper ,author has implemented two algorithm Adaline and Backpropion along withSVM for getting good detection rate and classification purpose.

Pradeepthi et al.[10] In this paper ,Author studied different classification algorithm and concluded that tree-based classifier are best and gives better accuracy for phishing URL detection. Also Author uses various Volume 3, Issue 7, September-October-2018 | <http://ijsrcseit.com> Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 398 features such aslexical features, URL based feature, network base features and domain based feature.

Luong et al. [11] proposed new technique to detect phishing website. In proposed method, Author used six heuristics that are primary domain, sub domain, path domain, page rank, and alexa rank, alexa reputation whose weight and values are evaluated. This approach gives 97 % accuracy but still improvement can be done by enhancing more heuristics.

Ahmad et al.[12] proposed three new features to improve accuracy rate for phishing website detection. In this paper, Author used both type of features as commonly known and new features for classification of phishing and non-phishing site. At the end author has concluded this work can be enhanced by using this novel features with decision tree machine learning classifiers.

### **3.2 REFERENCES**

Routhu Srinivasa Rao<sup>1</sup> , Alwyn Roshan Pais :Detection of phishing websites using an efficient feature-based machine learning framework :In Springer 2018. Volume 3, Issue 7, September-october-2018 | [http:// ijsrcseit.com](http://ijsrcseit.com)  
Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 399

Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : In ACM, 2018

Sudhanshu Gautam, Kritika Rani and Bansidhar Joshi : Detecting Phishing Websites Using Rule-Based Classification Algorithm: A Comparison : In Springer, 2018.

M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, Saira Gillani : A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms :In International Conference on Computational Science and Computational Intelligence IEEE ,2016.

Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto- Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE, 2017.

Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on

client- side using machine learning based approach :In Springer Science+Business Media, LLC,part of Springer Nature 2017

- [1] Bhagyashree E. Sananse, Tanuja K. Sarode : Phishing URL Detection: A Machine Learning and Web Mining-based Approach : In International Journal of Computer Applications,2015
- [2] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification Phishing Websites Based on URL : IEEE,2015
- [3] Priyanka Singh, Yogendra P.S. Maravi, Sanjeev Sharma : Phishing Websites Detection through Supervised Learning Networks : In IEEE,2015
- [4] Pradeepthi. K V and Kannan. A: Performance Study of Classification Techniques for Phishing URL Detection: In 2014 Sixth International Conference on Advanced Computing(ICoAC) IEEE,2014
- [5] Luong Anh Tuan Nguyen†, Ba Lam To† ,Huu Khuong Nguyen† and Minh Hoang Nguyen : Detecting Phishing Web sites: A Heuristic URL-Based Approach: In The 2013 International Conference on Advanced Technologies for Communications (ATC'13)
- [6] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013.

Mohammad et al. [13] proposed model that automatically extracts important features for phishing website detection without requiring any human intervention. Author has concluded in this paper that the process of extracting feature by their tool is much faster and reliable than any manual extraction

### **3.3PROBLEM STATEMENT DEFENETION**

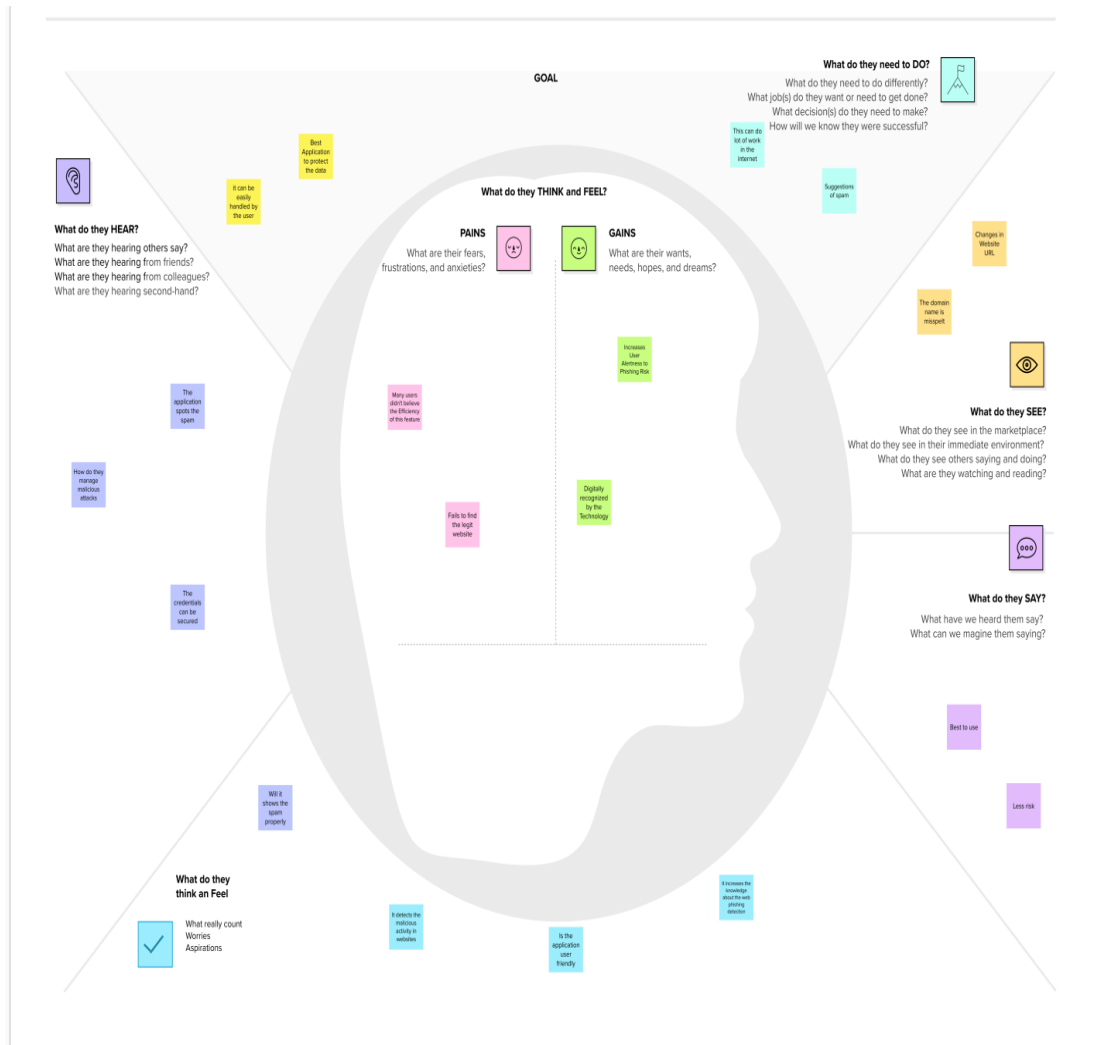
In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

Internet has dominated the world by dragging half of the world's population exponentially into the cyber world. With the booming of internet transactions, cybercrimes rapidly increased and with anonymity presented by the internet, Hackers attempt to trap the end-users through various forms such as phishing, SQL injection, malware, man-in-the-middle, domain name system tunnelling, ransomware, web trojan, and so on. Among all these attacks, phishing reports to be the most deceiving attack. Our main aim of this paper is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and concise model.

# CHAPTER 3

## IDEATION & PROPOSED SOLUTION

### Empathy Map Canvas





## 3.1 Ideation & Brainstorming

### Step-1: Team Gathering, Collaboration and Select the Problem Statement



#### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
🕒 1 hour to collaborate  
👤 2-8 people recommended

🗨️ Share template feedback



#### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes



#### Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.



#### Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.



#### Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

🔗 Open article →

1

#### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

How might we [your problem statement]?



#### Key rules of brainstorming

To run a smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

## Step-2: Brainstorm, Idea Listing and Grouping

2

### Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Tip

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

Niveetha P

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Prithika K

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Hemalatha

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Indumathi

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

3

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

Security

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Notification

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Feedbacks

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Additional Features

Implement a user authentication system

Design a user interface

Implement a user interface

Design a user interface

Implement a user interface

Design a user interface

Tip

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mind.

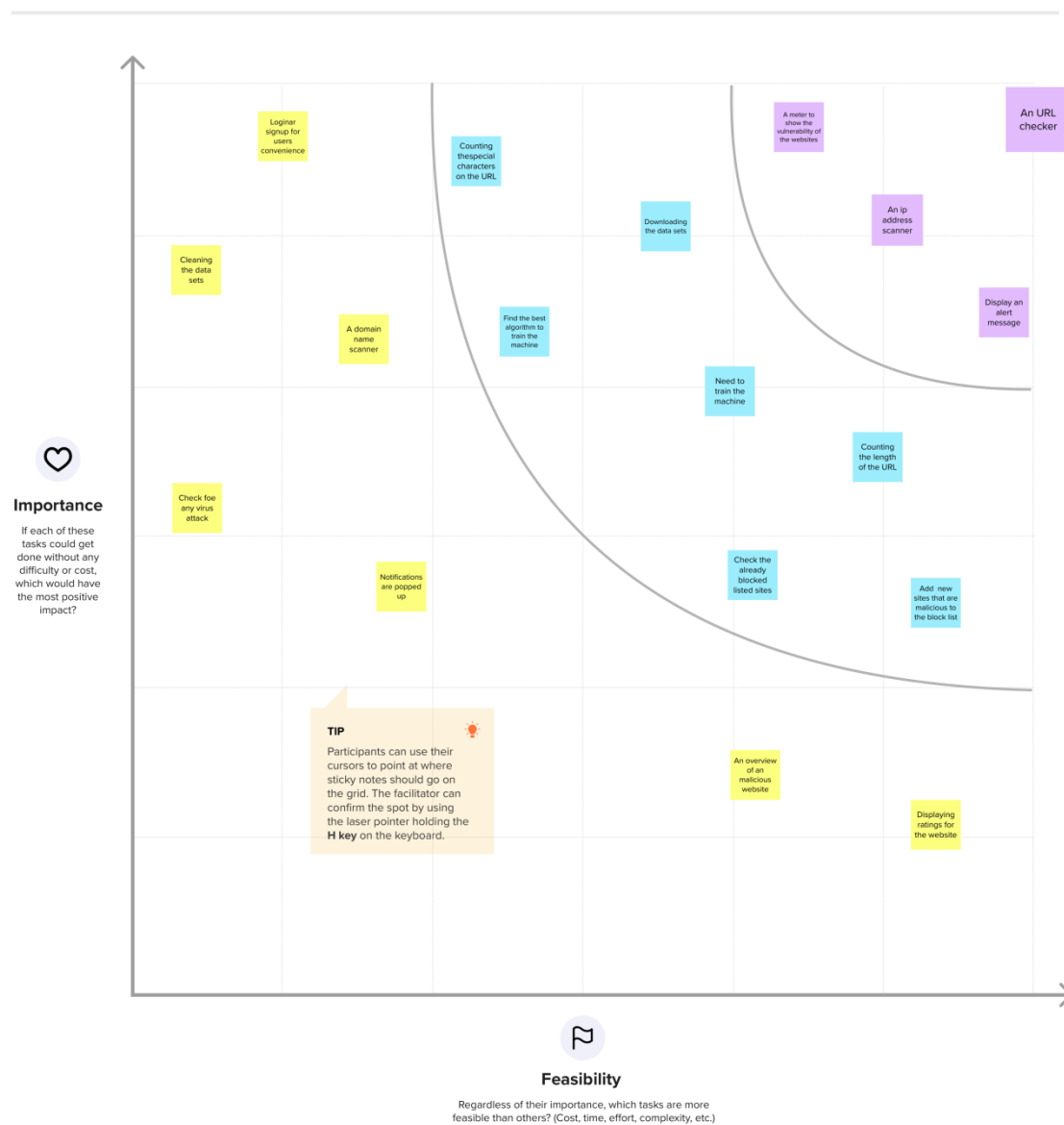
## Step-3: Brainstorm, Idea Listing and Grouping

4

### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



### Proposed Solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none"><li>• Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.</li><li>• It will lead to information disclosure and property damage.</li><li>• Large organizations may get trapped in different kinds of scams.</li></ul>
2.	Idea / Solution description	In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.
3.	Novelty / Uniqueness	The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

4.	Social Impact / Customer Satisfaction	The feasibility of implementing this idea is moderate neither easy nor tough because the system needs to satisfy the basic requirements of the customer as well as it should act as a bridge towards achieving high accuracy on
----	---------------------------------------	---

		predicting and analysing the detected websites or files to protect our customer to the fullest.
5.	Business Model (Revenue Model)	People buy subscription annually, to protect their files both locally and at remote location with the help of our cloud integrated flask app for web phishing detection.
6.	Scalability of the Solution	By implementing this system, the people can efficiently and effectively to gain knowledge about the web phishing techniques and ways to eradicate them by detection. This system can also be integrated with the future technologies

## 4.1 Problem Solution fit:

**Problem-Solution fit canvas 2.0** Purpose / Vision

<p><b>1. CUSTOMER SEGMENT(S)</b> <small>(Who is your customer?) i.e. working parents of 0-5 y.o. kids</small></p> <p>Ecommerce Consumers</p>	<p><b>6. CUSTOMER CONSTRAINTS</b> <small>(What constraints prevent your customers from taking action or limit their choices of solutions?) i.e. spending power, budget, no cash, network connection, available devices</small></p> <ul style="list-style-type: none"> <li>✓ Lack of awareness</li> <li>✓ Untraceable scam websites</li> <li>✓ Cloned websites</li> </ul>	<p><b>5. AVAILABLE SOLUTIONS</b> <small>(Which solutions are available to the customers when they face the problem, or need to get the job done? What have they used in the past? What pros &amp; cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking)</small></p> <ul style="list-style-type: none"> <li>✓ Existing web phishing detection websites</li> <li>✓ Word of Mouth</li> <li>✓ News coverage</li> <li>✓ Social Media</li> </ul>
<p><b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <small>(Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one, explore different sides)</small></p> <ul style="list-style-type: none"> <li>✓ Authentication of websites</li> <li>✓ Prevention of scams</li> </ul>	<p><b>9. PROBLEM ROOT CAUSE</b> <small>(What is the real reason that this problem arose? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations)</small></p> <ul style="list-style-type: none"> <li>✓ Greedy Scammers</li> <li>✓ Lack of awareness from customers</li> </ul>	<p><b>7. BEHAVIOUR</b> <small>(What does your customer do to address the problem and get the job done? i.e. directly related, find the right solar panel installer, calculate usage and benefits, indirectly associated, customers spend free time on volunteering work, i.e. Greenpeace)</small></p> <ul style="list-style-type: none"> <li>✓ Contacting Cybersecurity</li> <li>✓ Researching about website</li> <li>✓ Web community helpline</li> <li>✓ Reporting the site</li> </ul>
<p><b>3. TRIGGERS</b> <small>(What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news)</small></p> <ul style="list-style-type: none"> <li>✓ Reading about the E-Banking scams</li> <li>✓ Social Media</li> <li>✓ Past experiences</li> </ul> <p><b>4. EMOTIONS: BEFORE / AFTER</b> <small>(How do customers feel when they face a problem or a job and afterwards? i.e. lack, insecure &gt; confident, in control - use it in your communication strategy &amp; design)</small></p> <ul style="list-style-type: none"> <li>✓ Insecure &gt; Secure</li> <li>✓ Suspicious &gt; Trustworthy</li> </ul>	<p><b>10. YOUR SOLUTION</b> <small>(If you are working on an existing business, write down your current solution first, fit in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour)</small></p> <p>Verifies the genuineness of E-Banking websites/ Gateway</p>	<p><b>8. CHANNELS OF BEHAVIOUR</b> <b>8.1 ONLINE</b> <small>(What kind of actions do customers take online? Extract online channels from #7)</small></p> <ul style="list-style-type: none"> <li>✓ Researching website</li> <li>✓ Reporting the site</li> </ul> <p><b>8.2 OFFLINE</b> <small>(What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development)</small></p> <ul style="list-style-type: none"> <li>✓ Filing complaint with Bank</li> <li>✓ Contacting Cybersecurity</li> </ul>

Problem Solution Canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 license  
Created by Daria Negradina / Amaltama.com

**AMALTAMA**

## CHAPTER 4

### REQUIREMENT ANALYSIS

#### 4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Verifying input	User inputs an URL (Uniform Resource Locator) innecessary field to check its validation.
FR-2	Website Evaluation	Model evaluates the website using Blacklist and Whitelist approach
FR-3	Extraction and Prediction	It retrieves features based on heuristics and visual similarities. The URL is predicted by the model using Machine Learning methods such as Logistic Regressionand KNN.
FR-4	Real Time monitoring	The use of Extension plugin should provide a warningpop-up when they visit a website that is phished. Extension plugin will have the capability to also detectlatest and new phishing websites
FR-5	Authentication	Authentication assures secure site, secure processes and enterprise information security.



## 4.2 Non-functional Requirements:

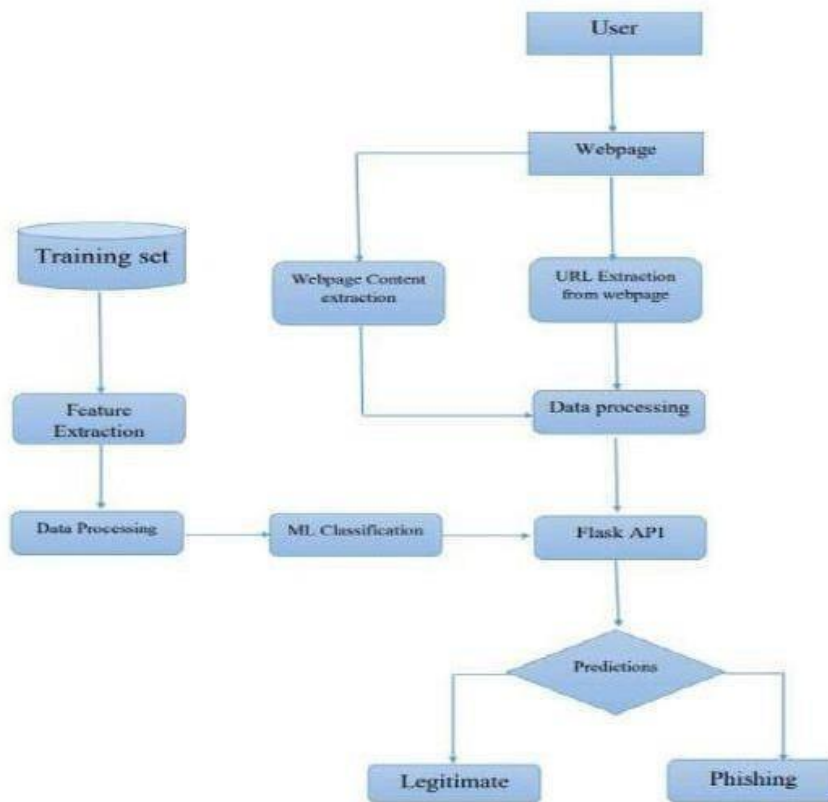
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-5	<b>Availability</b>	It represents the likelihood that a user will be able to access the system at a certain moment in time. While it can be represented as an expected proportion of successful requests, it can also be defined as a percentage of time the system is operational within a certain time period.
NFR-6	<b>Scalability</b>	It has access to the highest workloads that will allow the system to satisfy the performance criteria. There are two techniques to enable the system to grow as workloads increase: Vertical and horizontal scaling.
NFR-1	<b>Usability</b>	Analysis of consumers' product usability in the design process with user experience as the core may certainly help designers better grasp users' prospective demands in web phishing detection, behaviour, and experience.
NFR-2	<b>Security</b>	It guarantees that any data included within the system or its components will be safe from malware threats or unauthorised access. If you wish to prevent unauthorised access to the admin panel, describe the login flow and different user roles as system behaviour or user actions.
NFR-3	<b>Reliability</b>	It specifies the likelihood that the system or its component will operate without failure for a specified amount of time under prescribed conditions.
NFR-4	<b>Performance</b>	It is concerned with a measurement of the system's reaction time under various load circumstances.

## PROJECT DESIGN PHASE:

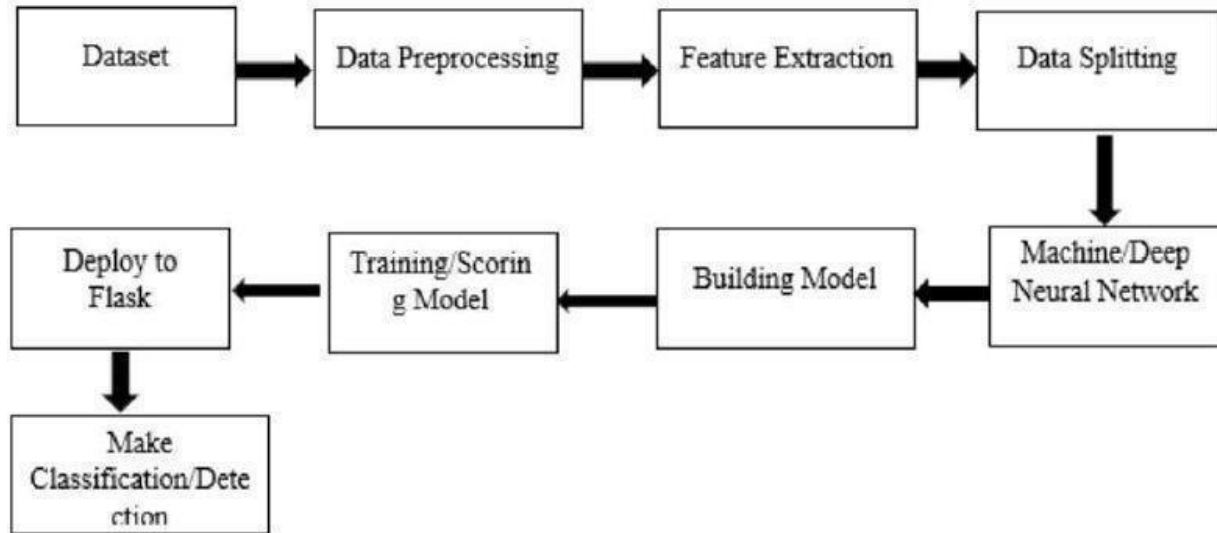
### 5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

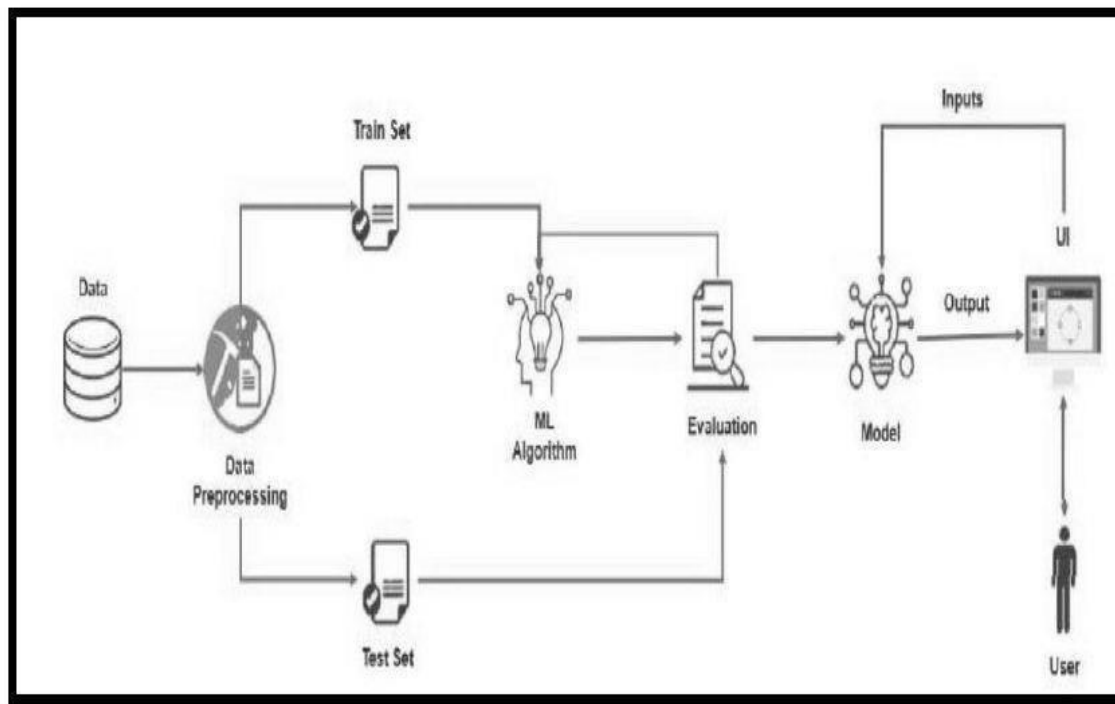


## 5.2 Solution and Technical Architecture

### Solution Architecture



### Technical Architecture: MODEL FOR WEB PHISHING DETECTION



## 5.3 USER STORIES

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Login	USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
		USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user i can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After i compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a User i can have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN	In this i can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here i will send all the model output to classifier in order to produce final result.	I this i will find the correct classifier for producing the result	Medium	Sprint-2

## CHAPTER 6

### PROJECT PLANNING & SCHEDULING

#### 6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	1	Medium	Hemalatha V
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High	Prithika K
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	2	High	Nivetha P
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	Indumathi C
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	1	Medium	Nivetha P
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	Prithika K
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	Indumathi C

## 6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

## 6.3 Reports from JIRA

	OCT	NOV	NOV	NOV
	24 25 26 27 28 29 30	31 1 2 3 4 5 6	7 8 9 10 11 12 13	14 15 16 17 18 19
Sprints	WPD Sprint 1	WPD Sprint 2	WPD Sprint 3	WPD Sprint 4
> WPD-8 User Input				
> WPD-9 Website Comparison				
> WPD-10 Feature Extraction				
> WPD-11 Prediction				
> WPD-12 Classifier				
> WPD-13 Announcement				
> WPD-14 Events				

# CHAPTER-7

## CODING & SOLUTION

### 7.1 Feature 1

```
#app.py
# importing required libraries

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

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

app = Flask( __name__ )

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

```
url = request.form["url"]
```



```

obj = FeatureExtraction(url)
x = np.array(obj.getFeaturesList()).reshape(1, 30)

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

if __name__=="_ main ":
    app.run(debug=True, port=2002)

```

## 7.2 Feature 2

```

#feature.py
import
ipaddress
import re
import urllib.request
from bs4 import
BeautifulSoupimport socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time

```

```
from dateutil.parser import parse as date_parse
```

```
from urllib.parse import urlparse
```

```
class
```

```
    FeatureExtraction:
```

```
        features = []
```

```
    def __init__(self, url):
```

```
        self.features =
```

```
        []self.url = url
```

```
        self.domain =
```

```
        """
```

```
        self.whois_response =
```

```
        """self.urlparse = """
```

```
        self.response = """
```

```
        self.soup = """
```

```
    try:
```

```
        self.response = requests.get(url)
```

```
        self.soup = BeautifulSoup(response.text,
```

```
        'html.parser')except:
```

```
        pass
```

```
    try:
```

```
        self.urlparse = urlparse(url)
```

```
        self.domain = self.urlparse.netloc
```

```
    except:
```

```
        pass
```

```
    try:
```

```
        self.whois_response =
```

```
        whois.whois(self.domain)except:
```

```
        pass
```

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

```
self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainUR
L()) self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags()
)
self.features.append(self.ServerFormHandler(
))self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding(
)) self.features.append(self.StatusBarCust())
```

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

```
self.features.append(self.LinksPointingToPage())
self.features.append(self.StatsReport())
```

# 1.UsingIp

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

# 2.longUrl

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

# 3.shortUrl

```
def shortUrl(self):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
```

```
'x|.co|prettylinkpro|.com|scrnch|.me|filoops|.info|vzturl|.com|qr|.net|lurl|.com|tweez|.me|v|.g  
d|tr|.im|link|.zip|.net', self.url)
```

```
    if match:  
        return -1  
  
    return 1
```

```
# 4.Symbol@
```

```
def
```

```
symbol(self):
```

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

```
# 5.Redirecting//
```

```
def
```

```
redirecting(self):
```

```
    if self.url.rfind('/') > 6:  
        return -1  
  
    return 1
```

```
# 6.prefixSuffix
```

```
def prefixSuffix(self):
```

```
    try:  
        match = re.findall('-',  
        self.domain)if match:  
            return -1  
        return  
    1  
    except:  
        return -1
```

```
# 7.SubDomains
```

```
def SubDomains(self):
```

```
dot_count = len(re.findall(".", self.url))
```

```
if dot_count == 1:
    return 1
elif dot_count == 2:
    return 0
return -1
```

# 8.HTTPS

```
def Hppts(self):
    try:
        https =
        self.urlparse.schemeif
        'https' in https:
            return 1
        return
    -1
except:
    return 1
```

# 9.DomainRegLen

```
def DomainRegLen(self):
    try:
        expiration_date =
        self.whois_response.expiration_datecreation_date =
        self.whois_response.creation_date try:
            if(len(expiration_date)):
                expiration_date =
                expiration_date[0]
        exce
        pt:
        pa
        ss
    try:
```



```
if(len(creation_date)):
    creation_date =
    creation_date[0]
except
    pt:
    pa
    ss
```

```

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

```

# 10. Favicon

```

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

```

# 11. NonStdPort

```

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

```

1

except:

return -1

# 12. HTTPSDomainURL

```
def HTTPSDomainURL(self):
```

```
    try:
```

```
        if 'https' in
```

```
            self.domain: return
```

```
            -1
```

```
        return
```

```
    1
```

```
    except:
```

```
        return -1
```

# 13. RequestURL

```
def RequestURL(self):
```

```
    try:
```

```
        for img in self.soup.find_all('img', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', img['src'])]
```

```
            if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
```

```
                success = success + 1
```

```
            i = i+1
```

```
        for audio in self.soup.find_all('audio', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
```

```
            if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
```

```
                success = success + 1
```

```
            i = i+1
```

```
        for embed in self.soup.find_all('embed', src=True):
```

```
            dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
```

```
            if self.url in embed['src'] or self.domain in embed['src'] or len(dots) ==
```

```
                1: success = success + 1
```

```
            i = i+1
```

```

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

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

```

#### # 14. AnchorURL

```

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

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

```

```

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

except:
    return -1

```

#### # 15. LinksInScriptTags

```
def LinksInScriptTags(self):
```

```

    try:
        i, success = 0, 0

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

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

    try:
        percentage = success / float(i) * 100

```

```
    if percentage < 17.0:
        return 1
    elif((percentage >= 17.0) and (percentage < 81.0)):
        return 0
    else:
        return
-1 except:
    return
0except:
    return -1
```

# 16. ServerFormHandler

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

# 17. InfoEmail

```
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?}", self.soap):
```

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

# 18. AbnormalURL

```
def AbnormalURL(self):
    try:
        if self.response.text ==
            self.whois_response: return 1
        else:
            return
    -1 except:
        return -1
```

# 19. WebsiteForwarding

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

# 20. StatusBarCust

```
def
    StatusBarCust(self
    ):try:
```



```
    if re.findall("<script>.+onmouseover.+</script>", self.response.text):
        return 1
    else:
        return
-1 except:
    return -1
```

#### # 21. DisableRightClick

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

#### # 22. UsingPopupWindow

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

#### # 23. IframeRedirection

```
def
IframeRedirection(self):
    try:
        if re.findall(r"<iframe>|<frameBorder>", self.response.text):
```

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

# 24. AgeofDomain

```
def
```

```
AgeofDomain(self):
```

```
    try:
```

```
        creation_date =
```

```
        self.whois_response.creation_date
```

```
        if(len(creation_date)):
```

```
            creation_date =
```

```
            creation_date[0]
```

```
    except:
```

```
        print:
```

```
        pass
```

```
        pass
```

```
    today = date.today()
```

```
    age = (today.year-creation_date.year) *
```

```
           \ 12+(today.month-
```

```
               creation_date.month)
```

```
    if age
```

```
        >= 6:
```

```
            return
```

```
            1
```

```
    return
```

```
-1
```

```
except:
```

```
    return -1
```

# 25. DNSRecording

def DNSRecording(self):

try:

creation\_date =

self.whois\_response.creation\_date

if(len(creation\_date)):

```

        creation_date =
creation_date[0]except:
    pass

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

```

# 26. WebsiteTraffic

```

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

```

# 27. PageRank

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

```

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

```

# 28. GoogleIndex

```

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

```

# 29. LinksPointingToPage

```

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

```

return -1

# 30. StatsReport

def StatsReport(self):

try:

url\_match = re.search(

'at\,ua\usa\,cc\baltazarpresentes\,com\,br\pe\,hu\esy\,es\hol\,es\sweddy\,com\myjino\,ru\96\,lt  
|ow\,ly', url)

ip\_address =

socket.gethostbyname(self.domain)ip\_match =

re.search('146\,112\,61\,108|213\,174\,157\,151|121\,50\,168\,88|192\,185\,217\,116|78\,46\,21  
1\,158|181\,174\,165\,13|46\,242\,145\,103|121\,50\,168\,40|83\,125\,22\,219|46\,242\,145\,98  
|'

'107\,151\,148\,44|107\,151\,148\,107|64\,70\,19\,203|199\,184\,144\,27|107\,151\,148\,108|10  
7\,151\,148\,109|119\,28\,52\,61|54\,83\,43\,69|52\,69\,166\,231|216\,58\,192\,225|'

'118\,184\,25\,86|67\,208\,74\,71|23\,253\,126\,58|104\,239\,157\,210|175\,126\,123\,219|141\  
.8\,224\,221|10\,10\,10\,10|43\,229\,108\,32|103\,232\,215\,140|69\,172\,201\,153|'

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

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

'216\,38\,62\,18|104\,130\,124\,96|47\,89\,58\,141|78\,46\,211\,158|54\,86\,225\,156|54\,82\,1  
56\,19|37\,157\,192\,102|204\,11\,56\,48|110\,34\,231\,42', ip\_address)if

url\_match:

return -1

elif

ip\_match:

return -1

return

1

except:

return 1

def

getFeaturesList(self)



```
:return self.features
```

# CHAPTER 8 TESTING

## 8.1 Test Cases

## TESTCASES REPORT

				Date	15-Nov-21								
				Team ID	PRJ2021MD42127								
				Project Name	Project - Web Phishing Detection								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requrite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1. Enter URL and click go 2.Type the URL 3.Verify whether it is processing or not.	<a href="https://phishing-shield.herokuapp.com/">https://phishing-shield.herokuapp.com/</a>	Should Display the Webpage	Working as expected	Pass		N		Hemalatha V
LoginPage_TC_002	UI	Home Page	Verify the UI elements is Responsive		1. Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	<a href="https://phishing-shield.herokuapp.com/">https://phishing-shield.herokuapp.com/</a>	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		Prishka K
LoginPage_TC_003	Functional	Home page	Verify whether the link is legitimate or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	<a href="https://phishing-shield.herokuapp.com/">https://phishing-shield.herokuapp.com/</a>	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		Indumathi C
LoginPage_TC_004	Functional	Home Page	Verify user is able to access the legitimate website or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate	<a href="https://phishing-shield.herokuapp.com/">https://phishing-shield.herokuapp.com/</a>	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Nisha P
LoginPage_TC_005	Functional	Home Page	Testing the website with multiple URLs		1. Enter URL ( <a href="https://phishing-shield.herokuapp.com/">https://phishing-shield.herokuapp.com/</a> ) and click go 2. Type or copy paste the URL to visit 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	1. <a href="https://www.github.io/website.com">https://www.github.io/website.com</a> 2. <a href="https://www.khask.edu.sg/secureinfo">https://www.khask.edu.sg/secureinfo</a> 3. <a href="https://www.google.com/delgata.com/">https://www.google.com/delgata.com/</a>	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Indumathi c

## 8.2 User Acceptance Testing

### UAT Execution & Report Submission

#### 1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Web Phishing Detection] project at the time of the release to User Acceptance Testing (UAT).

#### 2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	60

### 3. Test Case Analysis

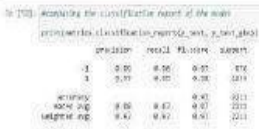

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

# CHAPTER 9

## RESULTS

### 9.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	<b>Classification Model:</b> <b>Gradient Boosting Classification</b> Accuracy Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

#### 1. METRICS:

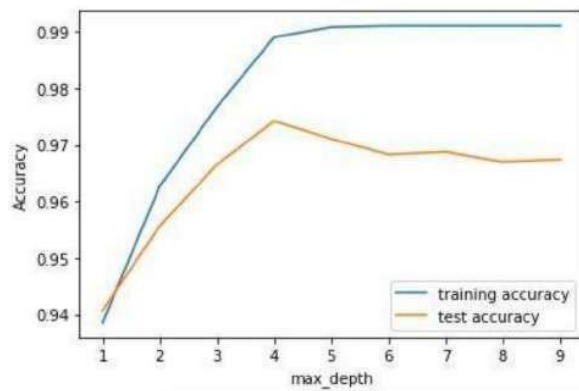
#### CLASSIFICATION REPORT:

In [52]: *#computing the classification report of the model*

```
print(metrics.classification_report(y_test, y_test_gbc))
```

	precision	recall	f1-score	support
-1	0.99	0.96	0.97	976
1	0.97	0.99	0.98	1235
accuracy			0.97	2211
macro avg	0.98	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

## PERFORMANCE :



Out[83]:

	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	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

## 2. TUNE THE MODEL – HYPERPARAMETER TUNING

```
In [58]: #HYPERPARAMETER TUNING  
grid.fit(X_train, y_train)
```

```
Out[58]: GridSearchCV  
  
GridSearchCV(cv=5,  
             estimator=GradientBoostingClassifier(learning_rate=0.7,  
                                                  max_depth=4),  
             param_grid={'max_features': array([1, 2, 3, 4, 5]),  
                        'n_estimators': array([ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130,  
140, 150, 160, 170, 180, 190, 200])})  
  
      estimator: GradientBoostingClassifier  
      GradientBoostingClassifier(learning_rate=0.7, max_depth=4)  
  
      GradientBoostingClassifier  
      GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
In [59]: print("The best parameters are %s with a score of %.2f"  
             % (grid.best_params_, grid.best_score_))
```

The best parameters are {'max\_features': 5, 'n\_estimators': 200} with a score of 0.97

## VALIDATION METHODS: KFOLD & Cross Folding

### Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

from scipy.stats import wilcoxon
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold

# Load the dataset
X = load_iris().data
y = load_iris().target

# Prepare models and select your CV method
model1 = GradientBoostingClassifier(n_estimators=100)
model2 = XGBClassifier(n_estimators=100)
kf = KFold(n_splits=20, random_state=None)
# Extract results for each model on the same folds
results_model1 = cross_val_score(model1, X, y, cv=kf)
results_model2 = cross_val_score(model2, X, y, cv=kf)
stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
stat
```

Out[78]: 95.0

### 5x2CV combined F test

```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                             estimator2=clf2,
                             X=X, y=y,
                             random_seed=1)

print('f-value:', f)
print('p-value:', p)

f-value: 1.727272727272733
p-value: 0.2840135734291782
```



## **CHAPTER -10**

### **Advantages of web phishing detection**

1. Improve on Inefficiencies of SEG and Phishing Awareness Training
2. It Takes a Load off the Security Team
3. It Offers a Solution, Not a Tool
4. Separate You from Your Competitors
5. This system can be used by many e-commerce websites in order to have goodcustomer relationships.
6. If internet connection fails this system will work

### **Disadvantages of web phishing detection**

1. All website related data will be stored in one place.
2. It is a very time-consuming process.

## **CHAPTER 11**

### **CONCLUSION**

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to tackle a wide assortment of classification issues, the procedure of finding the ideal structure is very difficult, and much of the time, this structure is controlled by experimentation. Our model takes care of this issue via computerizing the way toward organizing a neural system conspire; hence, on the off chance that we construct an enemy of phishing model and for any reasons we have to refresh it, at that point our model will encourage this procedure, that is, since our model will mechanize the organizing procedure and will request scarcely any client defined parameters.

## **CHAPTER-12**

### **FUTURE SCOPE**

There is a scope for future development of this project. We will implement this using advanced deep learning method to improve the accuracy and precision. Enhancements can be done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

## **CHAPTER-13**

### **Appendix:**

1. Application Building
2. Collection of Dataset
3. Data Pre-processing
4. Integration of Flask App with IBM Cloud
5. Model Building
6. Performance Testing
7. Training the model on IBM
8. User Acceptance Testing
9. Ideation Phase
10. Preparation Phase
11. Project Planning
12. Performance Testing
13. User Acceptance Testing

Project Link: <https://github.com/IBM-EPBL/IBM-Project-46363-1660746051>

Project Demo Link:

<https://drive.google.com/file/d/111LV9lcRUf-Vz4xnQGdiFDj8qZiMnT8q/view?usp=drivesdk>