



ANALYTICS OF HOSPITAL HEALTH CARE DATA

NALAIYA THIRAN PROJECT BASED LEARNING

On

**PROFESSIONAL READINESS FOR INNOVATION,
EMPLOYABILITY AND ENTREPRENEURSHIP**

A PROJECT REPORT

TEAM MENTOR: Mrs.KAVIPRIYA

TEAM LEADER: SATHIYA PRIYAN R 720819205045

TEAM MEMBERS: DHINA S 720819205011
NAVEEN S 720819205027
THANGAM S 720819205049

TEAM ID: PNT2022TMID10601

BATCH: B4-4M6E

YEAR: FINAL YEAR

**BACHELOR OF TECHNOLOGY
IN**

INFORMATION TECHNOLOGY

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY

Approved by AICTE, New Delhi, Accredited with 'A' Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai)

COIMBATORE – 641 032 - November 2022

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION 1.1 Project Overview 1.2 Purpose	4
2.	LITERATURE SURVEY 2.1 Existing problem 2.2 References 2.3 Problem Statement Definition	5
3.	IDEATION & PROPOSED SOLUTION 3.1 Empathy Map Canvas 3.2 Ideation & Brainstorming 3.3 Proposed Solution 3.4 Problem Solution fit	9
4.	REQUIREMENT ANALYSIS 4.1 Functional requirement 4.2 Non-Functional requirements	17
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	
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 13.1 Source Code 13.2 GitHub & Project Demo Link	

1. INTRODUCTION

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.

Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams.

This Guided Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.

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.

1.1 Project Overview

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- Applying different algorithms according to the datasets.
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

1.2 Project Purpose

- The purpose of Phishing Detection is detecting phishing domain names. which we want to classify as phishing or not, provide useful information to us.
- We can secure any devices using this project from unwanted Phishing websites.
- Using this project to identify during the transaction website legitimate or phishing.
- Eliminate the cyber threat risk level.
- Increase user alertness to phishing risk.

2.Literature Survey

2.1Existing Problem

Title :A Survey of Phishing Website Detection Systems

Author : Prachit Raut¹, Harshal Vengurlekar², Rishikesh Shete³

Abstract :Phishing URL is a widely used and common technique for cyber security attacks. Phishing is a cybercrime that tries to trick the targeted users into exposing their private and sensitive information to the attacker. The motive of the attacker is to gain access to personal information such as usernames, login credentials, passwords, financial account details, social networking data, and personal addresses. These private credentials are then often used for malicious activities such as identity theft, notoriety, financial gain, reputation damage, and many more illegal activities. This paper aims to provide a comprehensive and comparative study of various existing free service systems and research based systems used for phishing website detection. The systems in this survey range from different detection techniques and tools used by many researchers. The approach included in these researched papers ranges from Blacklist and Heuristic features to visual and content-based features. The studies presented here use advanced machine learning and deep learning algorithms to achieve better precision and higher accuracy while categorizing websites as phishing or benign. This article would provide a better understanding of the current trends and existing systems in the phishing detection domain.

Title:Phishing Detection using Machine Learning based URL Analysis:

Author : Arathi Krishna V, Anusree A, Blessy Jose

Abstract As we have moved most of our financial, work related and other daily activities to the internet, we are exposed to greater risks in the form of cybercrimes. URL based phishing attacks are one of the most common threats to the internet users. In this type of attack, the attacker exploits the human vulnerability rather than software flaws. It targets both individuals and organizations, induces them to click on URLs that look secure, and steal confidential information or inject malware on our system. Different machine learning algorithms are being used for the detection of phishing URLs, that is, to classify a URL as phishing or legitimate. Researchers are constantly trying to improve the performance of existing models and increase their accuracy. In this work we aim to review various machine learning methods used for this purpose, along with datasets and URL features used to train the machine learning models. The performance of different machine learning

algorithms and the methods used to increase their accuracy measures are discussed and analyzed. The goal is to create a survey resource for researchers to learn the current developments in the field and contribute in making phishing detection models that yield more accurate results.

Title : A survey on anti-phishing in websites:

Author : Robat D, Mukhter H, Shariful I, Abujarr.

Abstract : Phishing and fraud sites have been widespread on the internet in recent times, which's become a source of great concern and a serious cyber security problem, as internet fraudsters target sensitive data and personal information of users, especially the username and password. Numerous approaches have been proposed and used to prevent and reduce these phishing websites and attacks, and protect users and their privacy. In this paper, we categorized the present ant phishing approaches into two main classes: Content-based and Non-content-based. The content-based approach is also classified into URL content analysis and webpage content analysis. This helps in finding out numerous anti-phishing techniques and algorithms to choose the best approach in future contributions.

Title : A Survey of Phishing Attack Technique:

Author : Pratik Patil , Prof. P.R. Devale

Abstract : It is a crime to practice phishing by employing technical tricks and social engineering to exploit the innocence of unaware users. This methodology usually covers up a trustworthy entity so as to influence a consumer to execute an action if asked by the imitated entity. Most of the times, phishing attacks are being noticed by the practiced users but security is a main motive for the basic users as they are not aware of such circumstances. However, some methodologies are limited to look after the phishing attacks only and the delay in detection is mandatory. In this paper we emphasize the various techniques used for the detection of phishing attacks. We have also discovered various techniques for detection and prevention of phishing. Apart from that, we have introduced a new model for detection and prevention of phishing attacks.

Title : Phishing Detection: A Literature Survey:

Author: Mahmoud Khonji, Youssef Iraqi

Abstract : This article surveys the literature on the detection of phishing attacks. Phishing attacks target vulnerabilities that exist in systems due to the human factor. Many cyber-attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest

element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks. This paper aims at surveying many of the recently proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also presented, such as: detection, offensive defense, correction, and prevention, which we believe is critical to present where the phishing detection techniques fit in the overall mitigation process.

2.2 Reference

1. (Prachit Raut¹, Harshal Vengurlekar², Rishikesh Shete³ ^{1,2,3}Department of Computer Engineering, Vasantdada Patil Pratishthan's College of Engineering and Visual Arts, Mumbai, Maharashtra, India.).
2. (Arathi Krishna V, Anusree A, Blessy Jose, Karthika Anilkumar, Ojus Thomas Lee, Department of Computer Science and Engineering, College of Engineering Kidangoor, Kottayam, India.)
3. (Mahmoud Khonji, Youssef Iraqi, Senior Member, IEEE, and Andrew Jones.)
4. (Pratik Patil, Prof. P. R. Devale M Tech Student, Information Technology, BVUCOE, Pune, India¹ Professor, Information Technology, BVUCOE, Pune, India.)
5. (Robat D., Mukhter H., Shariful I., Abujarr S. 2019. Learning a Deep Neural Network for Predicting Phishing Website. PhD Thesis, Brac University, Bangladesh.)

2.3 Problem Statements Definition

Customer Problem Statement :

Phishing is a fraudulent technique that uses social and technological tricks to steal customer identification and financial credentials. Social media systems use spoofed e-mails from legitimate companies and agencies to enable users to use fake websites to divulge financial details like usernames and passwords. Hackers install malicious software on computers to steal credentials, often using systems to intercept username and passwords of consumers' online accounts. Phishers use multiple methods, including email, Uniform Resource Locators (URL), instant messages, forum postings, telephone calls, and text messages to steal user information. The structure of phishing content is similar to the original content and trick users to access the content in order to obtain their sensitive data. The primary objective of phishing is to gain certain personal information for financial gain or use of identity theft. Phishing attacks are causing severe economic damage around the world. Moreover, Most phishing attacks target financial/payment institutions and webmail, according to the Anti-Phishing Working Group (APWG) latest Phishing pattern studies.

Problem Statement (PS)

PS-1



PS-2



PS-3



Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	Internet user	Browse the internet	I identify a scam	An attacker masquerades as a reputable entity	Unsafe about my information that is shared over the network
PS-2	Enterprise user	Open emails in the cloudserver	I detect malicious protocols	They are not cryptographically signed.	Emails are unverified and third party intrusion
PS-3	Buyer and Seller	Buy or sell the products in e-commerce website	I identify a scam	They are not properly clone website identify by some image,text or features not working properly.	My credit card details are stolen and my email and password are leaked

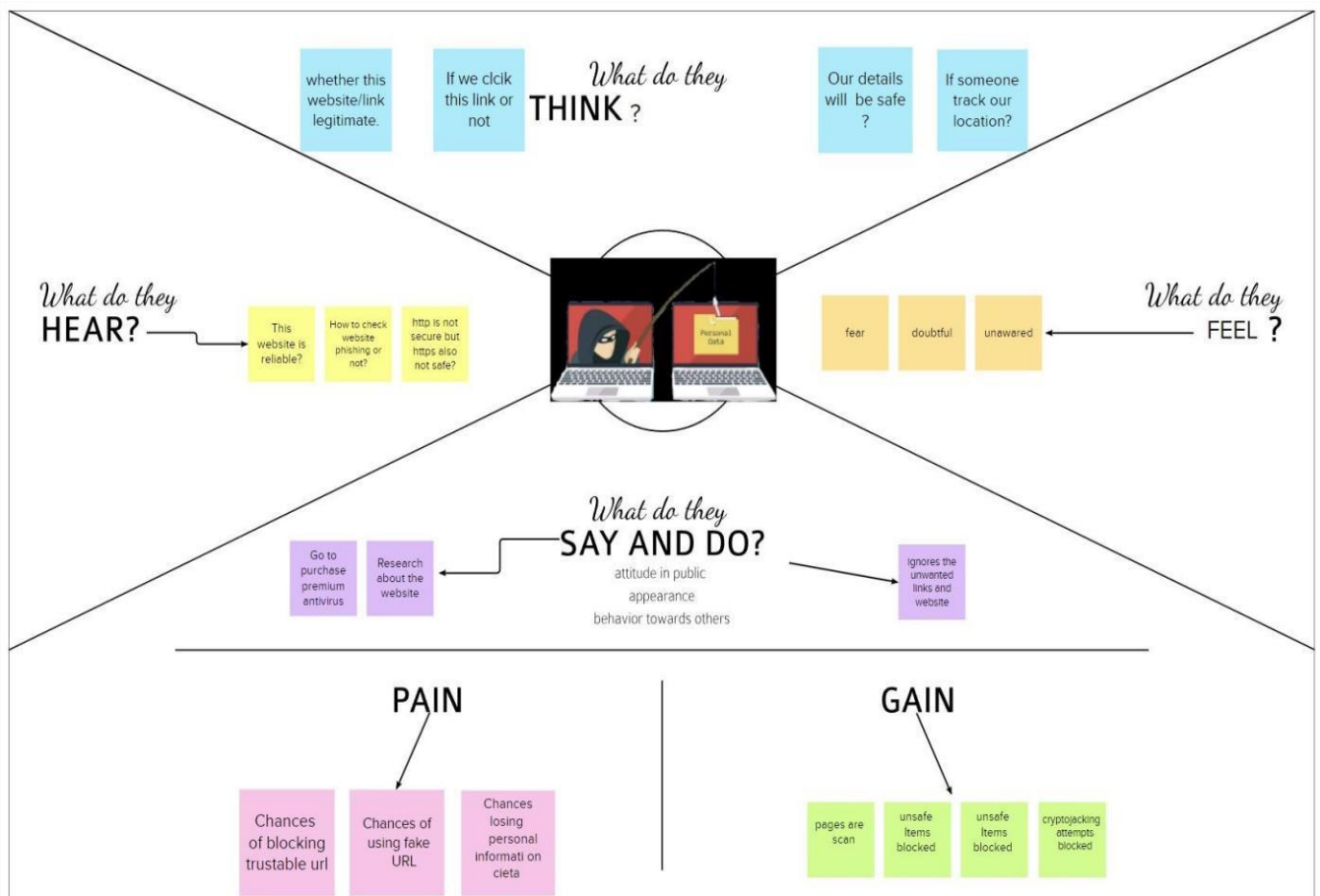
3.Ideation & Proposed Solution

3.1 Empathy Map Canvas

The Web Phishing Detection empathy map shows the visual that captures knowledge about a user's behaviors and attitudes. It is more useful for the teams better understand their users.

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

The below Empathy Map shows the user's behaviors and attitudes of Web Phishing Detection:




3.2 Ideation & Brainstorming

For the project Web Phishing Detection, Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

We as a team conducted Brainstorming which provided many solutions that led to problem solving.

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

Template



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

A

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

B

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

C

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

1) To implement a software solution to combat the problem of web phishing of users on the internet. We propose to implement a machine learning based model to classify the websites visited as Legitimate and Phishing sites. 2) We propose to implement an end-to-end ML application to capture the URL's in the address bar and use its feature to determine if its a legitimate site or a phishing site. 3) Progress for avoid phishing done by attackers which will be useful for an organization or any individuals.



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.

1

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

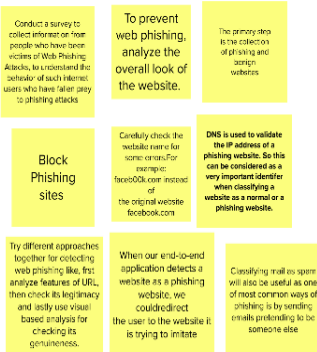
Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

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

R.Ajai Raaj



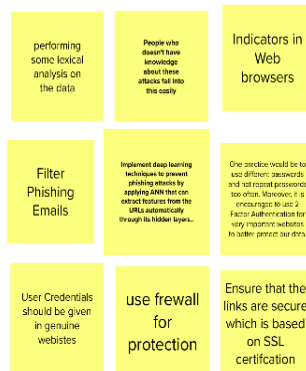
D.Durai pandi



S.Mohamed Rashik



P.Praveen



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

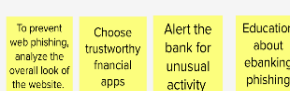
IDENTIFICATION OF PHISHING WEBSITE



TO AVOID PHISHING WEBSITE



PREVENTING THE E-BANKING PHISHING



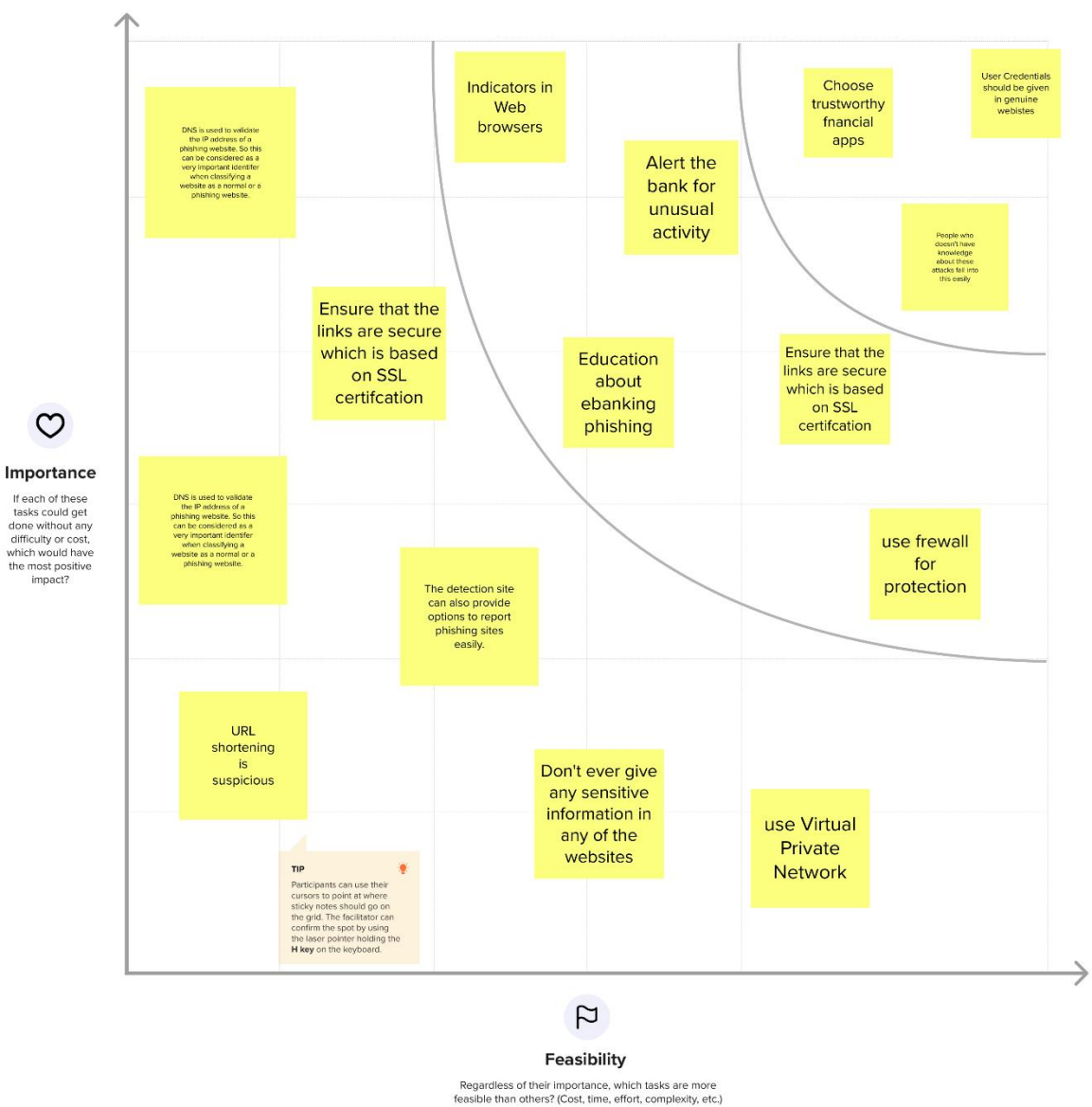
Step-3: Idea Prioritization

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



3.3Proposed Solution :

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Web phishing tends to steal a lots of information from the user during the online activities like online transaction like user important documents that has been attached to that websites . There are Multiple Types of Attacks happens here every day, but there is no auto detection Process through Machine Learning is achieved.
2.	Idea / Solution description	To use anti-phishing protection and anti-spam software to protect yourself. for example to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. 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. With the various user awareness of regular update of passwords to online account which makes the intruders hard to predict the password and never ever share your personal details and financial details over the internet because internet is the major and simple way for getting the user information so be aware of this.
3.	Novelty / Uniqueness	Machine learning technology consists of many algorithms which requires past data to make a decision or prediction of future data. Using this technique, algorithm will analyze various blacklisted and legitimate URL's and their features to accurately detect the phishing websites including zero-hour phishing websites.

4.	Social Impact / Customer Satisfaction	Phishing website has a list of effects on a business, including loss of money, loss of intellectual property, damage of reputation, and disruption of operational activities. Example: Facebook and Google between 2013 and 2015 Facebook and google were tricked out of \$100 million due to an extended phishing campaign. At present UBER had an social engineering based attack on one of their company employee's account where the attacker can able to access their internal cloud and etc. Customer Satisfaction: By using our web phishing detection website the user can check their websites by copy and paste the phishing URL. After knowing the result they can be completely safe from above mentioned impacts.
5.	Business Model (Revenue Model)	As long as phishing websites continue to operate, many more people and companies will suffer privacy leaks and data breaches or financial losses. However, the existing phishing detection method do not fully analyze the features of phishing and the performance and efficiency of the models only apply to certain limited datasets and further need to be improved to be applied to the real web environment.
6.	Scalability of the Solution	This project's performance rate will be high and it also provide many capabilities to the user without reducing its efficiency to detect the malicious websites. thus scalability of this project will be high .

3.4 Problem Solution Fit :

The Problem-Solution Fit for Web Phishing Detection is given below. The Problem Solution fit means that we have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps all the members of the project development recognize what would work and why.

Purpose:

- ❑ The Problem Solution fit solves complex problems in a way that fits the state of your customers.
- ❑ The Problem Solution fit succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
- ❑ The Problem Solution fit is used for marketing strategy with the right triggers and messaging.
- ❑ The Problem Solution fit increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- ❑ It is used to Understand the existing situation in order to improve it for your target group.

Problem – Solution Fit:

Project Title: Web Phishing Detection

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID10582

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS An user surfing through the internet for entertainment and social media. An enterprise user surfing through the internet for some information. An internet user who is willing to shop products online.	6. CUSTOMER CONSTRAINTS CC Customers is no knowledge about on phishing attack and websites. They don't know what to do after losing data. They don't know how to secure any devices from phishing attack	5. AVAILABLE SOLUTIONS AS Which solutions are available The already available solutions are blocking such phishing sites and by triggering a message to the customer about dangerous nature of the website. The blocking of phishing sites are not more effective as the attackers use a different/new site to steal potential data thus a AI/ML model. The place get knowledge about phishing is Ohphish website provided by EC-Council.
--	---	---

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P The phishing websites must be detected in a earlier stage . The user can be blocked from entering such sites for the prevention of such issues. The user can use any web extension used to find phishing website like McAfee Web Advisor.	9. PROBLEM ROOT CAUSE RC The hackers use new ways to cheat the naive users. Very limited research is performed on this part of the internet. Visiting some unwanted website and click link from unknown person.	7. BEHAVIOUR BE The option to check the legitimacy of the Websites is provided. Users get an idea what to do and more importantly what not to do.
--	--	---

Focus on J&P, tap into BE, understand RC

<div>Identify strong T & E M</div> <div> <p>3. TRIGGERS TR</p> <p>A trigger message can be popped warning the user about the site.</p> <p>Phishing sites can be blocked by the ISP and can show a “site is blocked” or “phishing site detected” message.</p> </div> <div> <p>4. EMOTIONS: BEFORE / AFTER EM</p> <p>How do customers feel when they face a problem or a job and afterwards?</p> <p>The customers feel lost and insecure to use the internet after facing such issues.</p> <p>Unwanted panicking of the customers is felt after encounter loss of potential data to such sites.</p> </div>	<div> <p>10. YOUR SOLUTION SL</p> <p>An option for the users to check the legitimacy of the websites is provided.</p> <p>This increases the awareness among users and prevents misuse of data,data theft etc.,</p> <p>Use extension for browser securing from phishing website .</p> <p>Always turn on your windows firewall and install some anti virus software.</p> </div>	<div> <p>8. CHANNELS of BEHAVIOUR CH</p> <p>8.1 ONLINE</p> <p>Customers tend to lose their data to phishing sites.</p> <p>8.2 OFFLINE</p> <p>Customers try to learn about the ways they get cheated from various resources viz., books, other people etc.,</p> </div> <div>Identify strong T & E M</div>
--	---	--

4.Solution Requirements (Functional & Non-functional)

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) in necessary 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 Regression and 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.

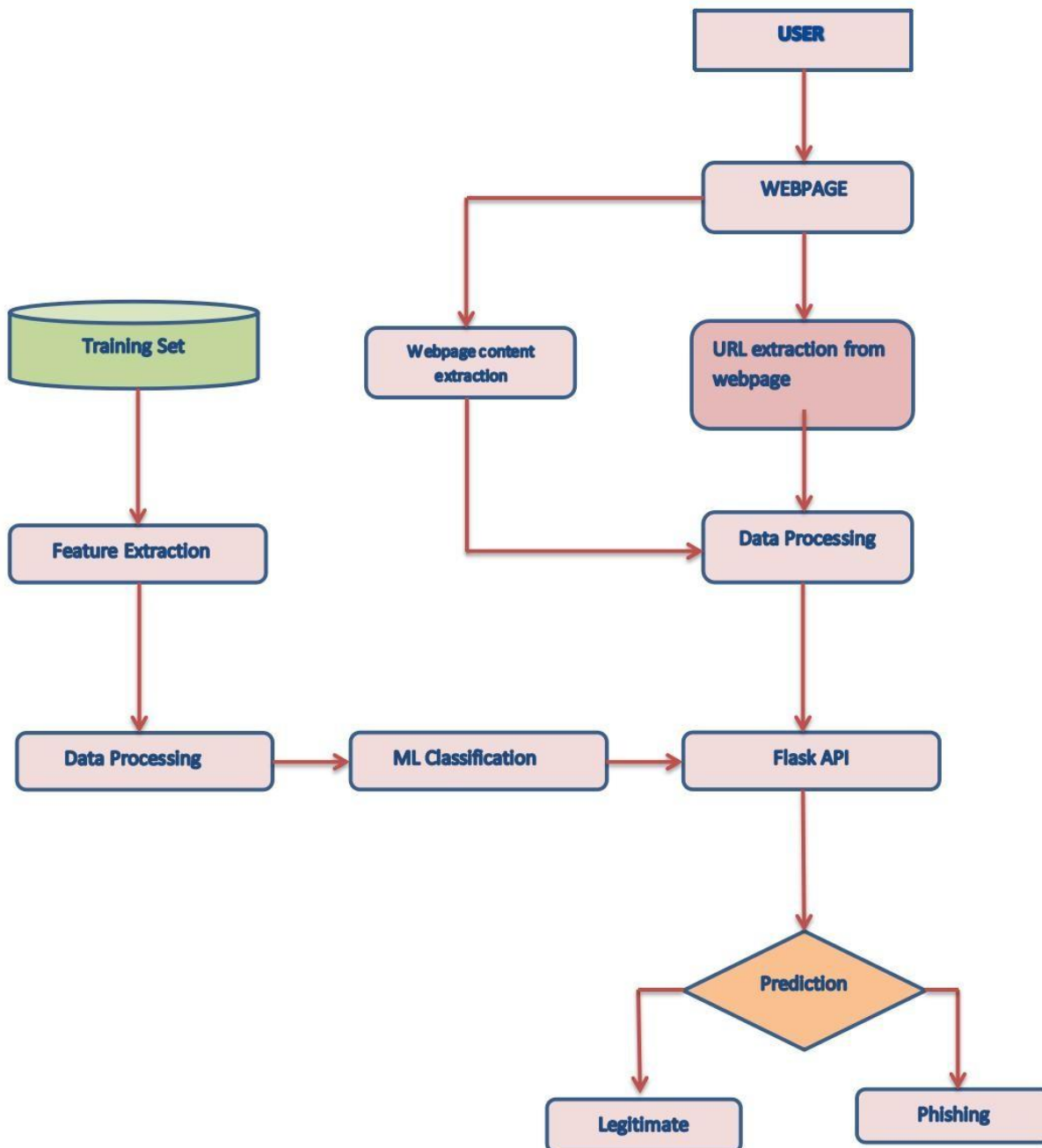
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Analysis of consumers' product usability in the design process with user experience as the core maycertainly help designers better grasp users' prospective demands in web phishing detection, behaviour, and experience.
NFR-2	Security	It guarantees that any data included within the system or its components will be safe from malwarethreats or unauthorized access. If you wish to prevent unauthorized access to the admin panel, describe the login flow and different user roles as system behaviour or user actions.
NFR-3	Reliability	It specifies the likelihood that the system or itscomponent will operate without failure for a specified amount of time under prescribed conditions.

NFR-4	Performance	It is concerned with a measurement of the system's reaction time under various load circumstances.
NFR-5	Availability	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	Scalability	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-7	Integrity	It guarantee that any data will be changes by only authorize people can make it ,like edit or delete.

5. Project Design

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.

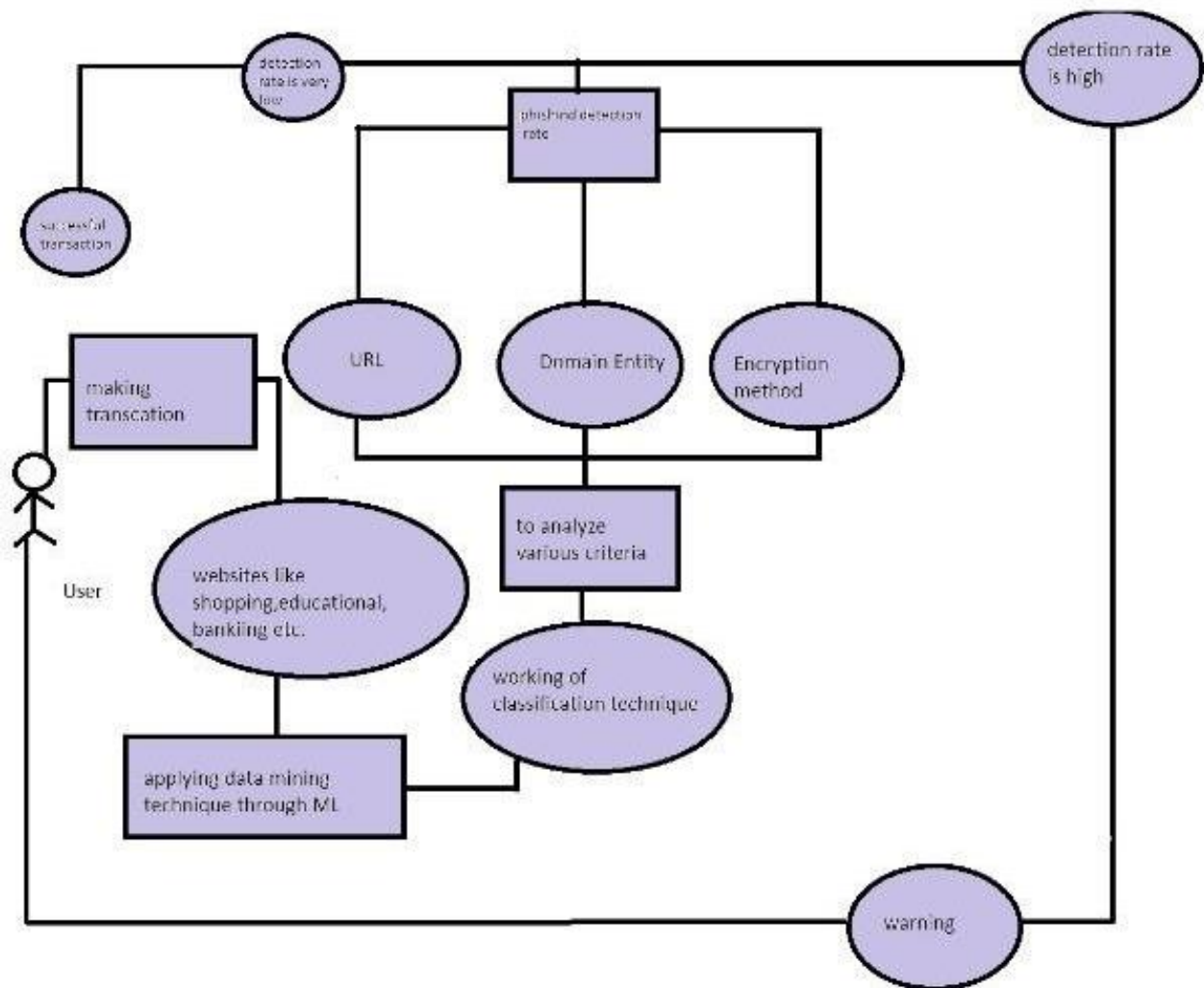


5.2 Solution & Technical Architecture

Web Phishing Detection solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- It is used to find the best tech solution to solve existing business problems.
- The solution architecture is used to describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- It defines the main features, development phases, and solution requirements for the Web Phishing Detection Project.
- It provides specifications according to which the solution for the Web Phishing Detection is defined, managed, and delivered.
- The below diagram shows the Solution Architecture Diagram of Web Phishing Detection Project.

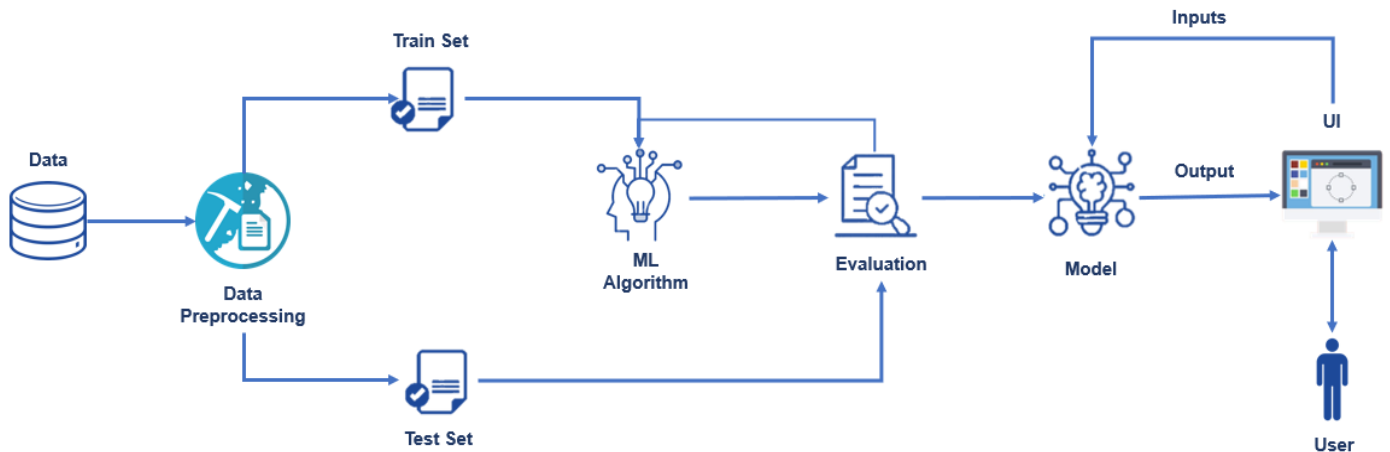
WEB PHISHING DETECTION SOLUTION ARCHITECTURE



Technical Architecture:

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
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Webuser)	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

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	AJAI RAAJ R
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High	MOHAMMED RASHIK S
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	2	High	MOHAMMED RASHIK S
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	PRAVEEN P
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	1	Medium	DURAI PANDI D
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or phishing site.	1	High	PRAVEEN P
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	AJAI RAAJ R

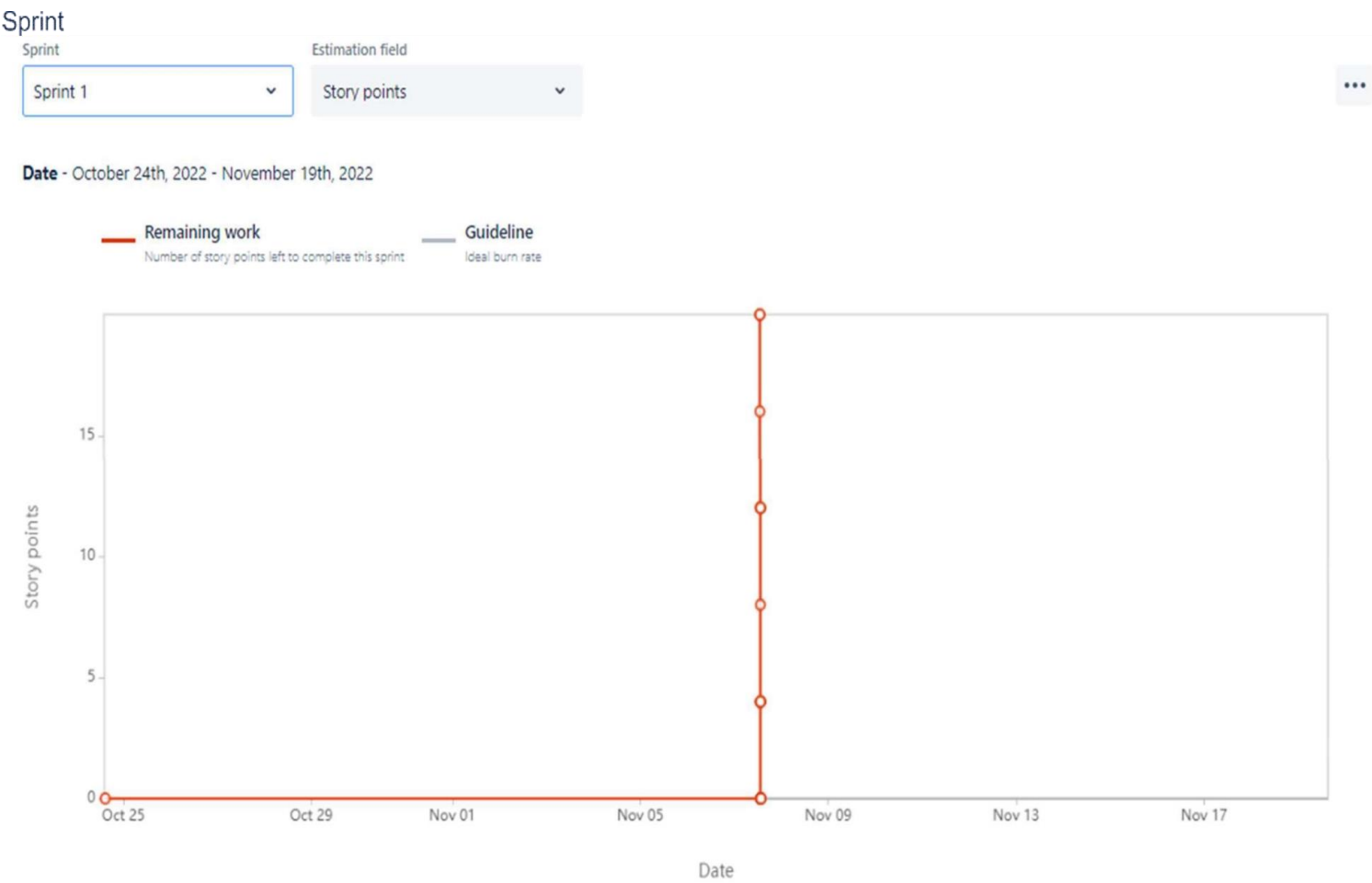
6.2 Sprint Delivery Schedule

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	2 Oct 2022	07 Nov 2022	20	8 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	20 Nov 2022

6.3 Reports from JIRA

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time for the project Web Phishing Detection. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



Sprint – 2:

Sprint

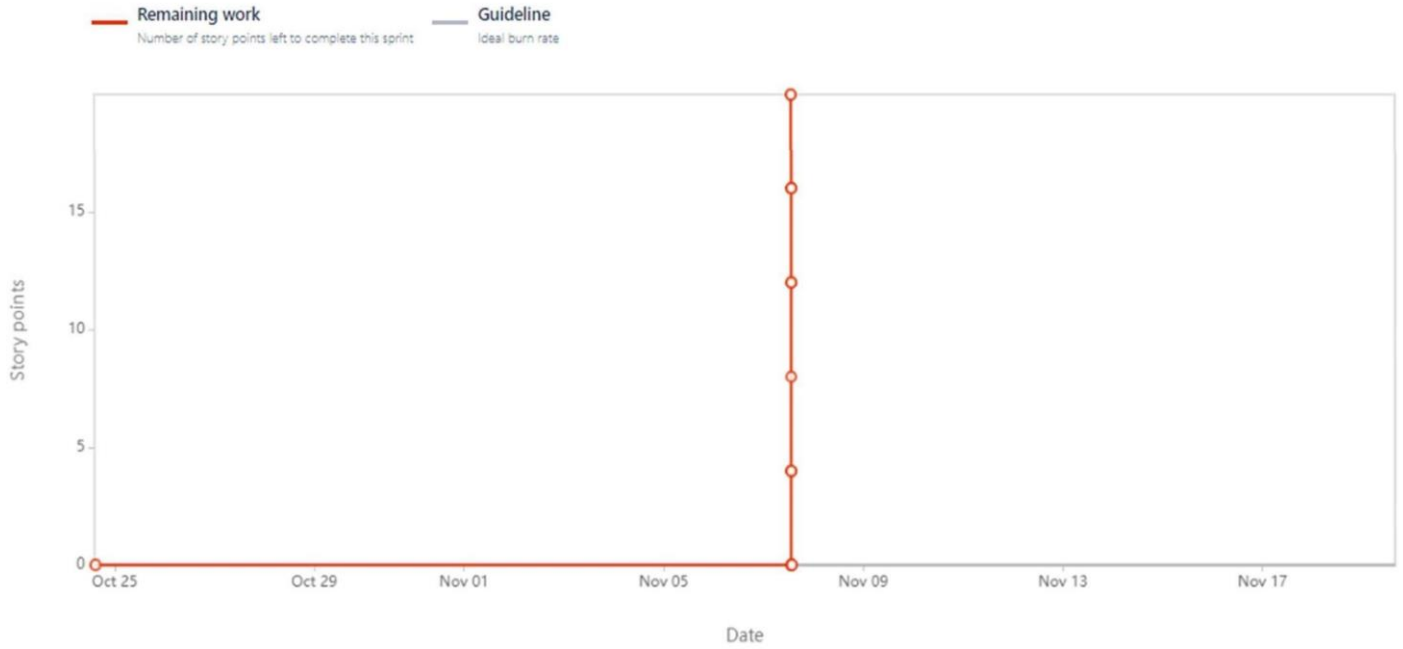
Sprint 2

Estimation field

Story points

...

Date - October 24th, 2022 - November 19th, 2022



Sprint

Sprint 3

Estimation field

Story points

...

Date - October 24th, 2022 - November 19th, 2022



Sprint-4:

Sprint

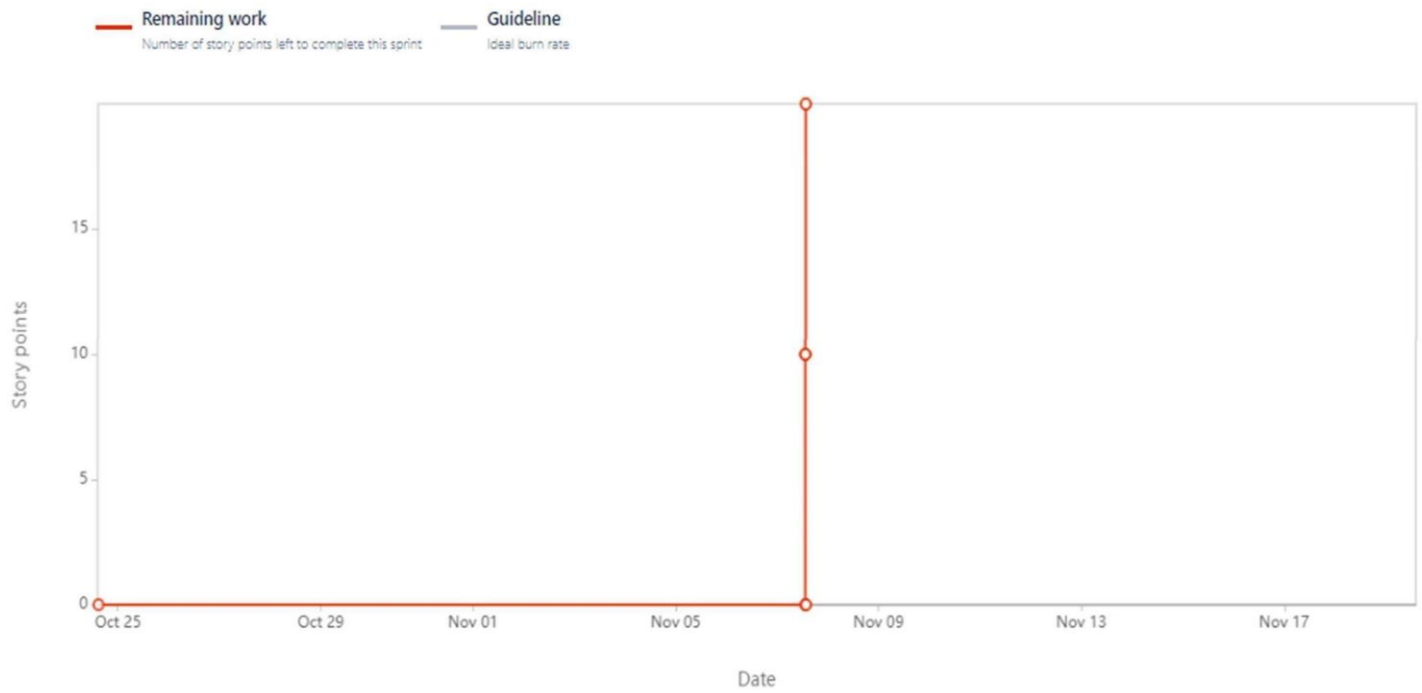
Sprint 4

Estimation field

Story points



Date - October 24th, 2022 - November 19th, 2022



Burnup Chart:

Sprint

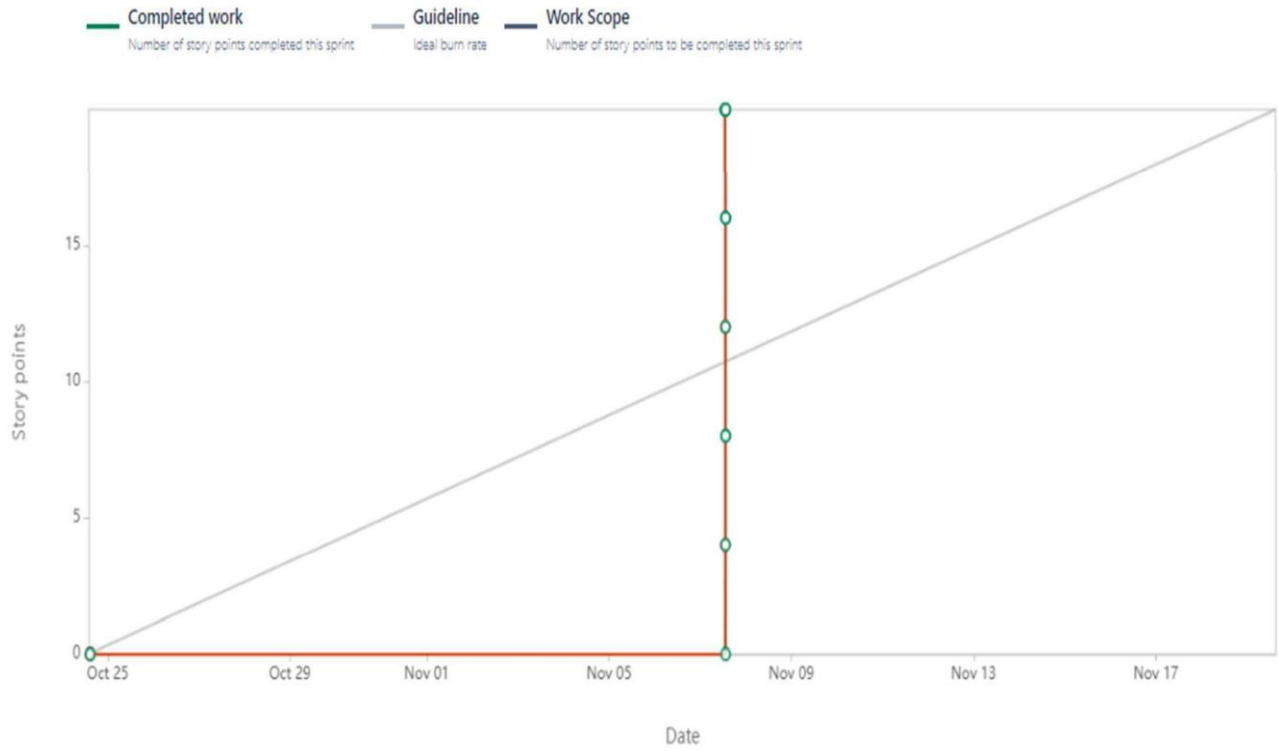
Sprint 1

Estimation field

Story points



Date - October 24th, 2022 - November 19th, 2022



7.CODING & SOLUTIONING

7.1 Feature 1 &7.2 Feature

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []

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

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

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

        try:
```

```

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

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

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

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

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

```

```

except:
    return 1

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

# 3.shortUrl
def shortUrl(self):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2\.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|twur\.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|k\.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|vztur\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.
        net',
        self.url)

    if match:
        return -1
    return 1

# 4.Symbol@
def symbol(self):
    if re.findall("@", self.url):
        return -1
    return 1

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

# 6.prefixSuffix
def prefixSuffix(self):
    try:

```

```

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

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

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

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

```

```

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

```

10. Favicon

```

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

```

11. NonStdPort

```

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

```

12. HTTPSDomainURL

```

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

```

13. RequestURL

```

def RequestURL(self):
    try:

```



```

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

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

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

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

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

# 14. AnchorURL
def AnchorURL(self):
    try:
        i, unsafe = 0, 0
        for a in self.soup.find_all('a', href=True):

```

```

        if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (
            url in a['href'] or self.domain in a['href']):
            unsafe = unsafe + 1
        i = i + 1

    try:
        percentage = unsafe / float(i) * 100
        if percentage < 31.0:
            return 1
        elif ((percentage >= 31.0) and (percentage < 67.0)):
            return 0
        else:
            return -1
    except:
        return -1

except:
    return -1

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

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

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

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

```

```

        else:
            return -1
    except:
        return 0
except:
    return -1

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

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

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

```

19. WebsiteForwarding

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

20. StatusBarCust

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

21. DisableRightClick

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

22. UsingPopupWindow

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

```
# 23. IframeRedirection
```

```
def IframeRedirection(self):
```

```
    try:
```

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

```
            return 1
```

```
        else:
```

```
            return -1
```

```
    except:
```

```
        return -1
```

```
# 24. AgeofDomain
```

```
def AgeofDomain(self):
```

```
    try:
```

```
        creation_date = self.whois_response.creation_date
```

```
    try:
```

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

```
            creation_date = creation_date[0]
```

```
    except:
```

```
        pass
```

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

```
    age = (today.year - creation_date.year) * 12 + (today.month - creation_date.month)
```

```
    if age >= 6:
```

```
        return 1
```

```
    return -1
```

```
    except:
```

```
        return -1
```

```
# 25. DNSRecording
```

```
def DNSRecording(self):
```

```
    try:
```

```
        creation_date = self.whois_response.creation_date
```

```
    try:
```

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

```
            creation_date = creation_date[0]
```

```
    except:
```

```
        pass
```

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

```
    age = (today.year - creation_date.year) * 12 + (today.month - creation_date.month)
```

```
    if age >= 6:
```

```

        return 1
    return -1
except:
    return -1

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

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

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

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

# 29. LinksPointingToPage

```

```

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

```

30. StatsReport

```
def StatsReport(self):
```

```
    try:
```

```
        url_match = re.search(
```

```
        'at\ua|usa\cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|ho\.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\.211\.158|181\.
        174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'
```

```
        '107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.
        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\.1
        9\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'
```

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

```
        '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.
        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
```


8. Testing

8.1 Test Cases

TESTCASES REPORT

				Date	16-Nov-22								
				Team ID	PNY20221MID10582								
				Project Name	Project - Web Phishing Detection								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisites	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO 1	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.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		Mohamed Rashik
LoginPage_TC_OO 2	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	https://phishing-shield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		Praveen
LoginPage_TC_OO 3	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	https://phishing-shield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		Durai pandi
LoginPage_TC_OO 4	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.	https://phishing-shield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Ajai raaj
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs		1. Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	1. https://avbalaiee.github.io/welcome_2_kajjad.com/ 2. https://www.kitce.edu 3. https://www.google.com/delights.com 4. https://www.google.com/delights.com 5. https://www.google.com/delights.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Mohamed Rashik

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

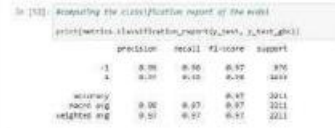

Exception Reporting	10	10	0	9
Final Report Output	10	0	0	10
Version Control	10	0	0	4

9.Result

9.1 Performance Metrics

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	Classification Model: Gradient Boosting Classification 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))
```

```

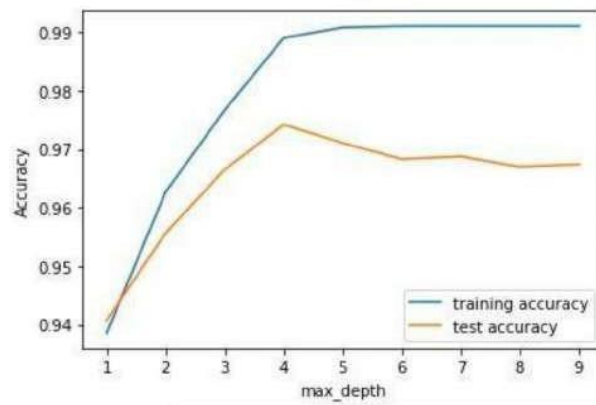
              precision    recall  f1-score   support

     -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
```

10.ADVANTAGES & DISADVANTAGES

Advantage and Disadvantage of This Project Detection:

No	Techniques Used	Advantages	Disadvantages
1	<i>Methods based on Bag-of-Words model</i>	-Build secure connection between user's mail transfer Agent (MTA) and mail user agent (MUA)	-Time consuming - huge number of features -consuming memory
2	<i>Compared multi Classifiers algorithms</i>	-Provide clear idea about the effective level of each classifier on phishing email detection	Non standard classifier
3	<i>hybrid system</i>	-High level of accuracy by take the advantages of many classifiers	-Time consuming because this technique has many layers to make the final result
4	<i>Classifiers Model-Based Features</i>	- High level of accuracy - create new type of features like Markov features	-huge number of features -many algorithm for classification which mean time consuming -higher cost -need large mail server and high memory requirement
5	<i>Clustering of Phishing Email</i>	-Fast in classification process	-Less accuracy because it depend on unsupervised learning , need feed continuously
6	Evolving Connectionist System (ECOS) for phishing email detection	fast ,less consuming memory, high accuracy, Evolving with time, online working	Need feed continuously

11. Conclusion

The proposed study emphasized the phishing technique in the context of classification, where phishing website is considered to involve automatic categorization of websites into a predetermined set of class values based on several features and the class variable. The ML based phishing techniques depend on website functionalities to gather information that can help classify websites for detecting phishing sites. The problem of phishing cannot be eradicated, nonetheless can be reduced by combating it in two ways, improving targeted anti-phishing procedures and techniques and informing the public on how fraudulent phishing websites can be detected and identified. To combat the ever evolving and complexity of phishing attacks and tactics, ML anti-phishing techniques are essential. Authors employed LSTM technique to identify malicious and legitimate websites. A crawler was developed that crawled 7900 URLs from AlexaRank portal and also employed Phishtank dataset to measure the efficiency of the proposed URL detector. The outcome of this study reveals that the proposed method presents superior results rather than the existing deep learning methods. A total of 7900 malicious URLs were detected using the proposed URL detector. It has achieved better accuracy and F1—score with limited amount of time. The future direction of this study is to develop an unsupervised deep learning method to generate insight from a URL. In addition, the study can be extended in order to generate an outcome for a larger network and protect the privacy of an individual.

11. FUTURE SCOPE

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique. In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features, Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers. Later on we will use at least two classifiers with a combination of the other to reach optimal accuracy. Additionally, we want to study different phishing techniques using Lexical highlights, organize focused highlights, content-related highlights, website-based highlights, and HTML and JavaScript website specific software that can help implement gadgets. Actually, we get highlights from URLs, then pass them through the different classifiers

12. APPENDIX

12.1 Source Code:

app.py

```
#importing required libraries
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction
file = open("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)

```

feature.py:

```

import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

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

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

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

```

```
except:
    pass

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

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

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

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

```

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

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

# 3.shortUrl
def shortUrl(self):
    match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|
cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.co
m|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|l
oopt\.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|your
ls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|t
weez\.me|v\.gd|tr\.im|link\.zip\.net', self.url)
    if match:
        return -1
    return 1

# 4.Symbol@
def symbol(self):
    if re.findall("@",self.url):
        return -1

```

```
    return 1
```

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

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

```
# 6.prefixSuffix
```

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

```
# 7.SubDomains
```

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

```
# 8.HTTPS
```

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

```
# 9.DomainRegLen
```

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

```

        pass
    try:
        if(len(creation_date)):
            creation_date = creation_date[0]
    except:
        pass

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

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

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

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

```

13. RequestURL

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

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

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

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

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

14. AnchorURL

```
def AnchorURL(self):
```



```

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

```

```

try:
    percentage = unsafe / float(i) * 100
    if percentage < 31.0:
        return 1
    elif ((percentage >= 31.0) and (percentage < 67.0)):
        return 0
    else:
        return -1
except:
    return -1

```

```

except:
    return -1

```

15. LinksInScriptTags

```

def LinksInScriptTags(self):

```

```

    try:
        i,success = 0,0

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

```

```

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

```

```

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

```

```

        return 0
    else:
        return -1
    except:
        return 0
except:
    return -1

```

16. ServerFormHandler

```

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

```

17. InfoEmail

```

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

```

18. AbnormalURL

```

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

```

19. WebsiteForwarding

```
def WebsiteForwarding(self):
```

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

```
# 20. StatusBarCust
```

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

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

```
# 21. DisableRightClick
```

```
def DisableRightClick(self):
```

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

```
# 22. UsingPopupWindow
```

```
def UsingPopupWindow(self):
```

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

```
# 23. IframeRedirection
```

```
def IframeRedirection(self):
```

```
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
```

```

        else:
            return -1
    except:
        return -1

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

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

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

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

# 26. WebsiteTraffic
def WebsiteTraffic(self):
    try:

```

```

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

```

27. PageRank

```

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

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

```

28. GoogleIndex

```

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

```

29. LinksPointingToPage

```

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

```

```

except:
    return -1

# 30. StatsReport
def StatsReport(self):
    try:
        url_match = re.search(

'at\,ua\usa\,cc\baltazarpresentes\,com\,br\pe\,hu\esy\,es\hol\,es\sweddy\,com\myj
ino\,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\,211\,158|181\,174\,165\,13|46\,242\,145\,103|121\,50\,168\,40|83\,
125\,22\,219|46\,242\,145\,98|'

'107\,151\,148\,44|107\,151\,148\,107|64\,70\,19\,203|199\,184\,144\,27|107\,15
1\,148\,108|107\,151\,148\,109|119\,28\,52\,61|54\,83\,43\,69|52\,69\,166\,231|2
16\,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\,17
0\,160\,61|213\,19\,128\,77|62\,113\,226\,131|208\,100\,26\,234|195\,16\,127\,1
02|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\,9
8\,255\,18|209\,99\,17\,27|'

'216\,38\,62\,18|104\,130\,124\,96|47\,89\,58\,141|78\,46\,211\,158|54\,86\,225\
,156|54\,82\,156\,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

```

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <center> <h1> IBM Project Based Learning </h1> </center>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="This website is develop for identify the
safety of url.">
  <meta name="keywords" content="phishing url,phishing,cyber
security,machine learning,classifïer,python">
  <meta name="author" content="Mohamed Rashik">

  <!-- Bootstrap -->
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css
"
  integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYYxFfc+NcPb
1dKGj7Sk" crossorigin="anonymous">

  <link href="static/styles.css" rel="stylesheet">
  <title>URL Detection</title>
</head>

<body>
  <center>  </center>

  <div class=" container">
    <div class="row">
      <div class="form col-md" id="form1">
        <h2>PHISHING URL DETECTION</h2>

        <br>
        <form action="/" method ="post">
          <input type="text" class="form__input" name ='url' id="url"
placeholder="Enter URL" required="" />
          <label for="url" class="form__label">URL</label>
          <button class="button" role="button" >Check here</button>
        </form>

      </div>
```

```

<div class="col-md" id="form2">

    <br>
    <h6 class = "right "><a href= {{ url }} target="_blank">{{ url
}}</a></h6>

    <br>
    <h3 id="prediction"></h3>
    <button class="button2" id="button2" role="button"
onclick="window.open('{{url}}')" target="_blank" >Still want to
Continue</button>
    <button class="button1" id="button1" role="button"
onclick="window.open('{{url}}')" target="_blank">Continue</button>
</div>
</div>
<br>
</div>

<!-- JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrC
XaRkfj"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMf
ooAo"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/k
R0JKI"
crossorigin="anonymous"></script>

<script>

    let x = '{{xx}}';
    let num = x*100;
    if (0<=x && x<0.50){
        num = 100-num;
    }

```



```

let txtx = num.toString();
if(x<=1 && x>=0.50){
    var label = "Website is "+txtx +"% safe to use...";
    document.getElementById("prediction").innerHTML = label;
    document.getElementById("button1").style.display="block";
}
else if (0<=x && x<0.50){
    var label = "Website is "+txtx +"% unsafe to use..."
    document.getElementById("prediction").innerHTML = label ;
    document.getElementById("button2").style.display="block";
}

</script>

</body>
<footer>
    <center> <p>© 2022 Mohamed Rashik </p> </center>
</footer>
</html>

```

Model.pkl

```

sklearn.ensemble._gb GradientBoostingClassifier
n_estimators 6 learning_rate 0.1 loss log_loss
criterion friedman_mse min_samples_split min_samples_leaf
min_weight_fraction_leaf max_features min_impurity_decrease
random_state max_leaf_nodes 6 warm_start validation_fraction
n_iter_no_change feature_names_in_ numpy.core.multiarray
_reconstruct numpy ndarray UsingIP LongURL ShortURL Symbol@
Redirecting// PrefixSuffix- SubDomains HTTPS
DomainRegLen Favicon NonStdPort HTTPSDomainURL
RequestURL AnchorURL LinksInScriptTags ServerFormHandler
InfoEmail AbnormalURL WebsiteForwarding StatusBarCust
DisableRightClick UsingPopupWindow IframeRedirection
AgeofDomain DNSRecording WebsiteTraffic PageRank GoogleIndex
LinksPointingToPage StatsReport n_features_in_ n_classes
n_classes_ sklearn.ensemble._gb_losses BinomialDeviance
max_features_ sklearn.dummy DummyClassifier strategy
prior sparse_output_ n_outputs_ class_prior_
sklearn_version estimators_ sklearn.tree._classes
DecisionTreeRegressor splitter best numpy.random._pickle
__randomstate_ctor 6 bit_generator class_weight
sklearn.tree._tree Tree node_count 6 left_child right_child
feature threshold impurity n_node_samples
weighted_n_node_samples train_score_ n_estimators_

style.css:

```

```

        *,
*::after,
*::before {
    margin: 0;
    padding: 0;
    box-sizing: inherit;
    font-size: 62,5%;
}
.image {
    width: 500px;
    height: 500px;
}
.image-contain {
    object-fit: contain;
    object-position: center;
}

.image-cover {
    object-fit: cover;
    object-position: center;
}
body {
    padding: 10% 5%;
    background: #0f2027;
    background: linear-gradient(to right,#2c5364, #203a43,
##55FFFF);
    justify-content: center;
    align-items: center;
    height: 100vh;
    color: #fff;
}

.form__label {
    font-family: 'Roboto', sans-serif;
    font-size: 1.2rem;
    margin-left: 2rem;
    margin-top: 0.7rem;
    display: block;
    transition: all 0.3s;
    transform: translateY(0rem);
}

.form__input {
    top: -24px;
    font-family: 'Roboto', sans-serif;
    color: #333;
    font-size: 1.2rem;
    padding: 1.5rem 2rem;

```

```

border-radius: 0.2rem;
background-color: rgb(255, 255, 255);
border: none;
width: 75%;
display: block;
border-bottom: 0.3rem solid transparent;
transition: all 0.3s;
}

.form__input:placeholder-shown + .form__label {
  opacity: 0;
  visibility: hidden;
  -webkit-transform: translateY(+4rem);
  transform: translateY(+4rem);
}

.button {
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, #fff,
#f8eedb);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-
ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto
Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe
UI Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px
rgba(81,41,10,0.2);
}

```

```
.button:active {
  background-color: #f3f4f6;
  box-shadow: -1px 2px 5px rgba(81,41,10,0.15),0px 1px 1px
  rgba(81,41,10,0.15);
  transform: translateY(0.125rem);
}
```

```
.button:focus {
  box-shadow: rgba(72, 35, 7, .46) 0 0 0 4px, -6px 8px 10px
  rgba(81,41,10,0.1), 0px 2px 2px rgba(81,41,10,0.2);
}
```

```
.main-body{
  display: flex;
  flex-direction: row;
  width: 75%;
  justify-content:space-around;
}
```

```
.button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245,
  174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-
  ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto
  Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe
  UI Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
```

```

    box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px
    rgba(81,41,10,0.2);
    display: none;
}

.button2{
    appearance: button;
    background-color: transparent;
    background-image: linear-gradient(to bottom, rgb(252, 162,
162), #ee3737);
    border: 0 solid #e5e7eb;
    border-radius: .5rem;
    box-sizing: border-box;
    color: #482307;
    column-gap: 1rem;
    cursor: pointer;
    display: flex;
    font-family: ui-sans-serif,system-ui,-apple-system,system-
ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto
Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe
UI Symbol","Noto Color Emoji";
    font-size: 100%;
    font-weight: 700;
    line-height: 24px;
    margin: 0;
    outline: 2px solid transparent;
    padding: 1rem 1.5rem;
    text-align: center;
    text-transform: none;
    transition: all .1s cubic-bezier(.4, 0, .2, 1);
    user-select: none;
    -webkit-user-select: none;
    touch-action: manipulation;
    box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px
    rgba(81,41,10,0.2);
    display: none;
}

.right {
    right: 0px;
    width: 300px;
}

@media (max-width: 576px) {
    .form {
        width: 100%;
    }
}
.abc{

```

```

    width: 50%;
}

```

integrate.ipynb:

```
#importing required libraries
```

```

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

```

```

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

```

```
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
```

```

API_KEY = "cWGD5yTjEpEGtqPpvHPDBElN5eXFS7eh2JRDyUWhySMW"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
    API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

```

```

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}

```

```
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)
```

```
    payload_scoring = {"input_data": [{"field":
```

```

[["UsingIP","LongURL","ShortURL","Symbol@","Redirecting//","PrefixSuffix-
","SubDomains","HTTPS","DomainRegLen","Favicon","NonStdPort","HTTPSDomainU
RL","RequestURL","AnchorURL","LinksInScriptTags","ServerFormHandler","Info
Email","AbnormalURL","WebsiteForwarding","StatusBarCust","DisableRightClic
k","UsingPopupWindow","IframeRedirection","AgeofDomain","DNSRecording","We
bsiteTraffic","PageRank","GoogleIndex","LinksPointingToPage","StatsReport"
]], "values": [[1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,-1,-1,1,1,1,0,1,1,1,1,-1,-
1,-1,-1,1,0,1]]]}}

```

```

        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/084b5c52-f617-40ef-a0e8-
3e6cf79ae447/predictions?version=2022-11-06', json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        print("Scoring response")
        predictions=response_scoring.json()
    #print(predictions)
        pred=print(predictions['predictions'][0]['values'][0][0])
        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=2020)

```

13.2 GitHub & Project Demo Link:

<https://github.com/IBM-EPBL/IBM-Project-32541-1660210601.git>

