WEB PHISHING DETECTION

DONE BY:

Team leader: Noble A
Team Member: Tharun J
Team Member: Vishva R
Team Member: Sathyaseelan S

1.INTRODUCTION

1.1 PROJECT OVERVIEW

The criminals, who want to obtain sensitive data, first create unauthorized replicas of a real website and e-mail. The e-mail will be created using logos and slogans of a legitimate company. The nature of website creation is one of the reasons that the Internet has grown so rapidly as a communication medium. Phisher then send the "spoofed" e-mails to as many people as possible in an attempt to lure them into the scheme. When these e-mails are opened or when a link in the mail is clicked, the consumers are redirected to a spoofed website, appearing to be from the legitimate entity. We discuss the methods used for detection of phishing Web sites based on URL importance properties.

Phishing has been accounted for many fraudulent incidents on the internet in the recent years, and it is showing no sign of stopping anytime soon. So, what is phishing? It is a term that is used to describe a malicious individual or a group of individuals who scam users. This is done by sending emails or creating web pages that are designed to collect an individual’s online credentials, credit card details or other login information’s. The concept of detecting phishing websites is usually done by looking through a huge database or a directory that contains all the malicious sites that has been logged by internet users or community members. An effective way for end users to benefit from phishing detection is by having the option to use an extension plugin that works on real time, as it gives them real time indication of what they are surfing and as well as if they are safe while browsing.

1.2 PROJECT PURPOSE

The main purpose of the project is to detect the fake or phishing websites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials. We are using machine learning algorithms to safeguard the sensitive data and to detect the phishing websites who are trying to gain access on sensitive data.

This research mainly will focus on implementing machine learning in JavaScript for it to run on a browser as an extension since JavaScript does not have much library support towards Machine Learning and also to keep in mind of the users' machines performance. This approach should be made with the intention of having it lite in order to achieve the capability to allow as much users as possible to use it.

Random forest classifier for this project will be trained traditionally based on the phishing dataset 2 using Python scikit, and parameters of this model will then be exported in a JSON format to be used together with JavaScript.

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels.

Typically a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.

The main reason is the lack of awareness of users. But security defenders must take precautions to prevent users from confronting these harmful sites. Preventing these huge costs can start with making people conscious in addition to building strong security mechanisms which are able to detect and prevent phishing domains from reaching the user

2. LITERATURE SURVEY

2.1 Existing Problem

Cyber criminals use phishing emails because it's easy, cheap and effective. Email addresses are easy to obtain, and emails are virtually free to send. With little effort and cost, attackers can quickly gain access to valuable data. Those who fall for phishing scams may end up with malware infections (including ransomware), identity theft and data loss.

The data that cybercriminals go after includes personal identifiable information (PII)—like financial account data, credit card numbers and tax and medical records—as well as sensitive business data, such as customer names and contact information, proprietary product secrets and confidential communications.Cybercriminals also use phishing attacks to gain direct access to email, social media and other accounts or to obtain permissions to modify and compromise connected systems, like point-of-sale terminals and order processing systems.

Many of the biggest data breaches, like the headline-grabbing 2013 Target breach, start with a phishing email. By using a seemingly innocent email, cybercriminals can gain a small foothold and build on it. in order to begin the development of a google chrome browser extension, it should emphasize on the ability to alert and warn the users if they accidentally visited a phishing webpage. This chrome extension will also be developed keeping in mind that, it should not have any 3rd party servers or API present to call services as this gives a narrow path for hackers to target users browsing pattern. Lastly, this extension plugin will also provide an instantaneous detection service that warns users as they view a phishing website, just so they avoid entering any confidential information before it is too late.

2.2 References

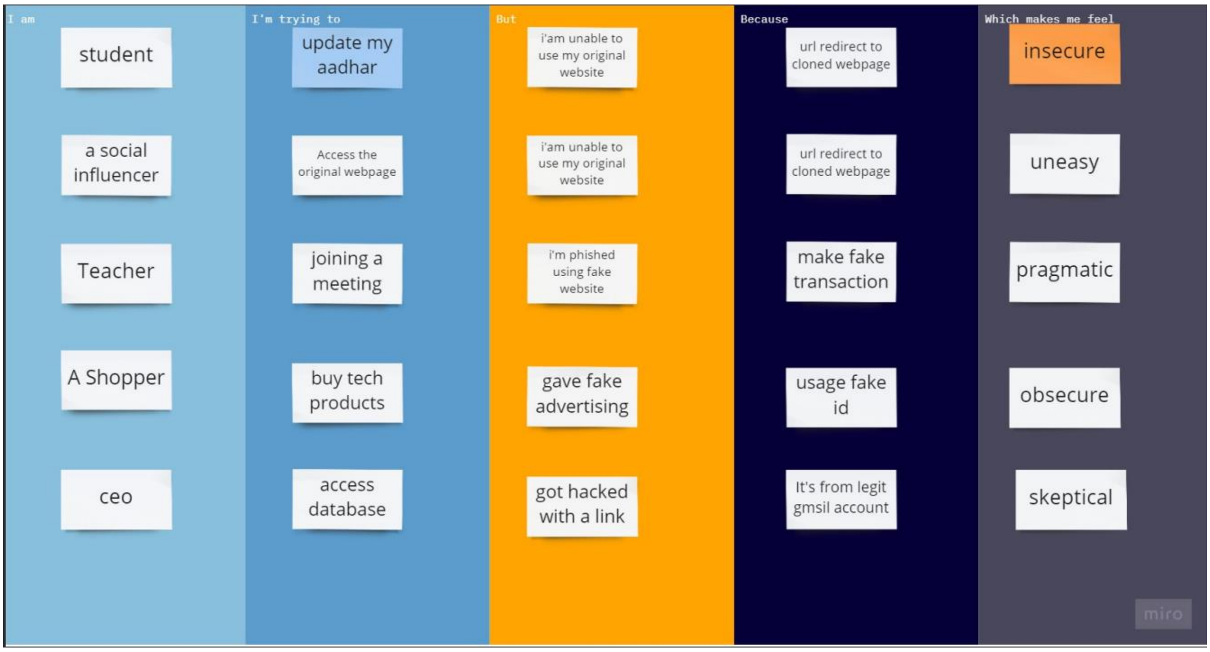
Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels.

Typically a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.

Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer’s defense systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization’s logos and other legitimate contents. In this article I explain: phishing domain (or Fraudulent Domain) characteristics, the features that distinguish them from legitimate domains, why it is important to detect these domains, and how they can be detected using machine learning and natural language processing techniques.

By : Ebubekir Bubar ,Ajay , Christian

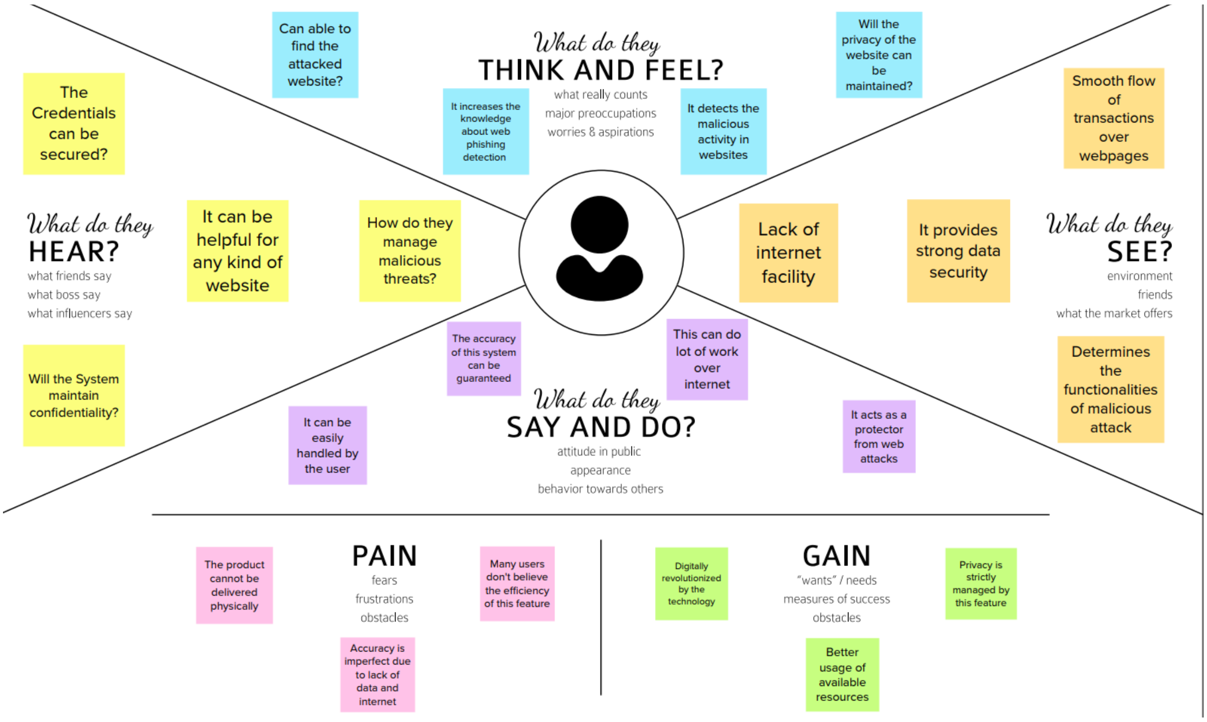
2.3 Problem Statement Definition



I am	I'm trying to	But	Because	Which makes me feel
student	update my aadhar	i'am unable to use my original website	url redirect to cloned webpage	insecure
a social influencer	Access the original webpage	i'am unable to use my original website	url redirect to cloned webpage	uneasy
Teacher	joining a meeting	i'm phished using fake website	make fake transaction	pragmatic
A Shopper	buy tech products	gave fake advertising	usage fake id	obscure
ceo	access database	got hacked with a link	It's from legit gmsil account	skeptical

3.IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations

Can able to find the attacked website?

It increases the knowledge about web phishing detection

It detects the malicious activity in websites

Will the privacy of the website can be maintained?

What do they
HEAR?
what friends say
what boss say
what influencers say

The Credentials can be secured?

It can be helpful for any kind of website

How do they manage malicious threats?

Will the System maintain confidentiality?

What do they
SEE?
environment
friends
what the market offers

Smooth flow of transactions over webpages

Lack of internet facility

It provides strong data security

Determines the functionalities of malicious attack

What do they
SAY AND DO?
attitude in public
appearance
behavior towards others

The accuracy of this system can be guaranteed

This can do lot of work over internet

It can be easily handled by the user

It acts as a protector from web attacks

PAIN
fears
frustrations
obstacles

The product cannot be delivered physically

Many users don't believe the efficiency of this feature

Accuracy is imperfect due to lack of data and internet

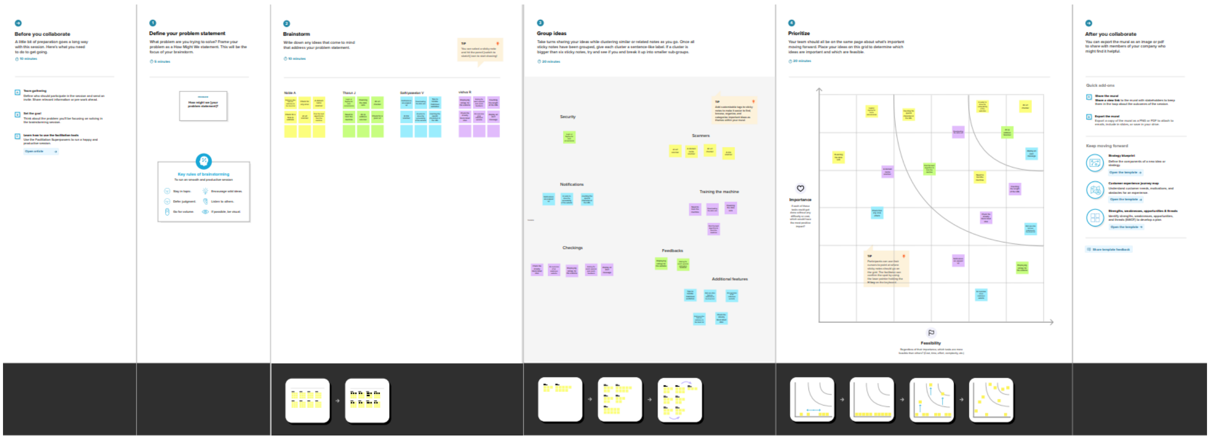
GAIN
"wants" / needs
measures of success
obstacles

Digitally revolutionized by the technology

Privacy is strictly managed by this feature

Better usage of available resources

3.2 Ideation and Brainstorm



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

How might we [your problem statement]?

Key rules of brainstorming

To run an smooth and productive session

Stay in topic. Encourage wild ideas.

Defer judgment. Listen to others.

Go for volume. If possible, be visual.

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!

Nithi A

Tharun J

Sathiyaseelan V

vishva R

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 and break it up into smaller sub-groups.

20 minutes

TIP

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

Security

Scanners

Notifications

Training the machine

Checkings

Feedbacks

Additional features

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

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural** Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural** Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint Define the components of a new idea or strategy.

Open the template →

Customer experience journey map Understand customer needs, motivations, and obstacles for an experience.

Open the template →

Strengths, weaknesses, opportunities & threats Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.

Open the template →

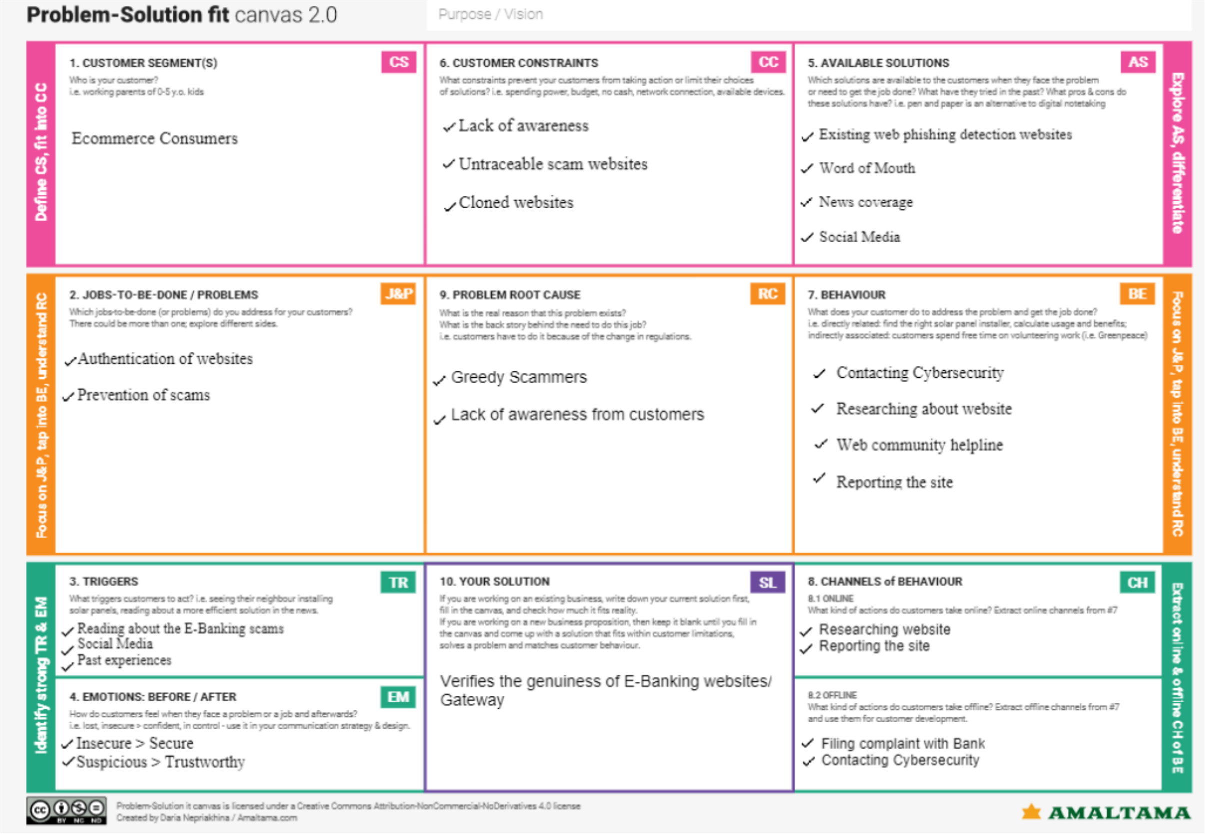
Share template feedback

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Many banking websites, and the anonymous links for malicious content, request users to submit private information like usernames, passwords, credit card numbers, and more. Phishing websites are this kind of online banking website. One of the many security risks to web services on the Internet is web phishing.
2.	Idea / Solution description	1. Address Bar based Features 2. Abnormal Based Features 3. HTML and JavaScript Based Features 4. Domain Based Feature. are the components are used for detection of phishing attack
3.	Novelty / Uniqueness	The proposed solution uses novel methods to identify relevant features by factoring in attributes like page rank, in-degree, presence of HTTPS token, and others. Through the features obtained, different dataset sizes and distributions are tested for analyzing and detecting optimal hyperparameters.
4.	Social Impact / Customer Satisfaction	The website is created with an opinion such that people are not only able to distinguish between legitimate and fraudulent website, but also become aware of the mal-practices. They can stay away from the people trying to exploit one's personal information, like email address, password, debit card numbers, credit card details, CVV, bank account numbers, and the list goes on.
5.	Business Model (Revenue Model)	Micro web frameworks like flask can be used to create a REST-based

		web application and web services that users may use to conduct reliable and secure online transactions through safe e-commerce websites and identify the illegal links. Based on membership levels, different levels of security strictness and multiple volumes of secure e-commerce websites would be offered.
6.	Scalability of the Solution	The proposed website's functionality can be turned into an API so that other e-banking and e-commerce portals can utilize it to provide their users with safer and more effective solutions. It could be further extended for other security concerns such as audio recording, video recording, location tracking, virus attacks, and more with safer and more effective solutions.

3.3 Problem Solution Fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Ecommerce Consumers

6. CUSTOMER CONSTRAINTS CC

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.

✓ Lack of awareness

✓ Untraceable scam websites

✓ Cloned websites

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

✓ Existing web phishing detection websites

✓ Word of Mouth

✓ News coverage

✓ Social Media

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

✓ Authentication of websites

✓ Prevention of scams

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
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.

✓ Greedy Scammers

✓ Lack of awareness from customers

7. BEHAVIOUR BE

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)

✓ Contacting Cybersecurity

✓ Researching about website

✓ Web community helpline

✓ Reporting the site

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

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

✓ Reading about the E-Banking scams
✓ Social Media
✓ Past experiences

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

✓ Insecure > Secure
✓ Suspicious > Trustworthy

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill 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.

Verifies the genuiness of E-Banking websites/ Gateway

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

✓ Researching website
✓ Reporting the site

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

✓ Filing complaint with Bank
✓ Contacting Cybersecurity

Extract online & offline CH of BE

AMALTAMA

4.REQUIREMENT ANALYSIS:

4.1 Functional Requirement

FR NO.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Input	User inputs an URL in required field to check its validation.
FR-2	Website Comparison	Model compares the websites using Blacklist and Whitelist approach.
FR-3	Feature extraction	After comparing, if none found on comparison then it extracts feature using heuristic and visual similarity approach.
FR-4	Prediction	Model predicts the URL using Machine Learning algorithms such as Logistic Regression, KNN
FR-5	Classifier	Model sends all output to classifier and produces final result.
FR-6	Announcement	Model then displays whether website is a legal site or a phishing site.
FR-7	Events	This model needs the capability of retrieving and displaying accurate result for a website

4.2 . **Non-functional Requirements:**

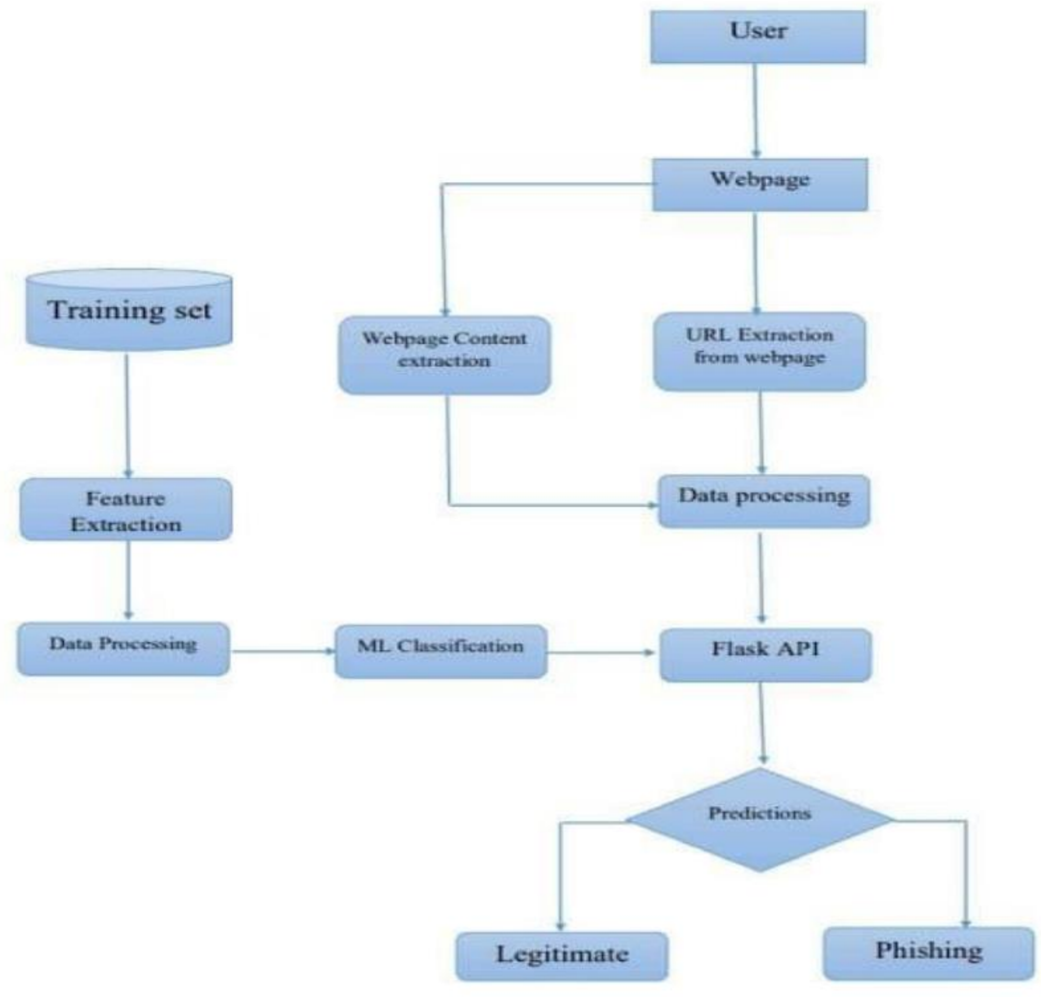
Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	Any URL must be accepted for detection
NFR-2	Security	Alert message must be send to the users to enable secure browsing.
NFR-3	Reliability	The Phishing websites must detected accurately and the results must be reliable
NFR-4	Performance	The performance and interface must be user friendly
NFR-5	Availability	Anyone must be able to register and login
NFR-6	Scalability	It must be able to handle increase in the number of users.

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.

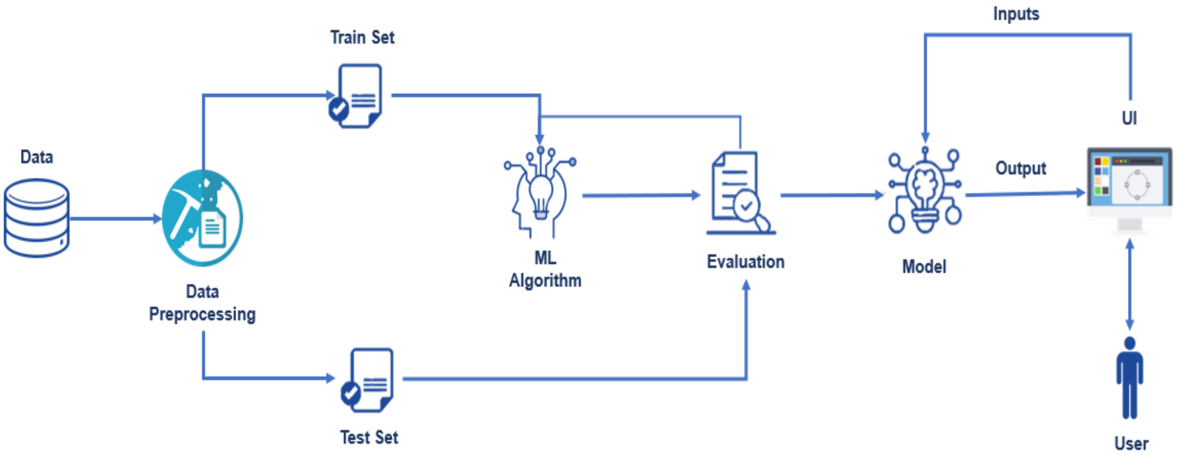


Use the below template to list all the user stories for the product.

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 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 via Facebook		Low	Sprint-2
		USN-4	As a user, I can register for the application via 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 (Web user)	User input	USN-1	As a user i can input the particular	I can go access the website	High	Sprint-1

			URL in the required field and waiting for validation.	without any problem		
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 have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning	In this i can have correct prediction on the particular algorithms	High	Sprint-1

5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptation Criteria	Priority	Release
Customer (web user)	Website	USN – 1	Some, times the user need to check the website to check particular URL is safe or not	Website has easy to use and responsive.	High	Sprint-1
	Alter Notification	USN – 2	If enter into some malicious link, notification has to be sent to me	Received notification in mobile or to my mail ID	Low	Sprint-1
	Blocking	USN – 3	Whenever the link is not safe to enter, it should block to use to me.	I can register and access the dashboard with Facebook login.		Sprint-2
	Allowing	USN – 4	If I wish to use that website then it should also allow me to enter into that website			Sprint-1
	Login	USN – 5	As a user, I can log into the application by entering email & password	The phishing website has to be determined correctly.	High	Sprint-1
	DSHBOARD					

Customer (web view)	User input	USN - 1	As a user I can enter the required URL in the box while awaiting validation	I can access the website without any problem	High	
Customer care executive	Feature extraction	USN - 1	In the event that nothing is discovered during comparison, we can extract features	As a user I can have comparison between websites for security	High	
			using a heuristic and a visual similarity technique.			
administrator	Prediction	USN - 1	The model will use machine learning algorithms like a logistic regression and KNN model to forecast the URL of the websites.	I can accurately forecast the specific algorithms in this way	High	
	classifier	USN - 2	To create the final product. I will now feed all of the model output to classifier.	I will use this to identify the appropriate classifier for generating the outcome	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning And Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers,research publications etc.	16 OCTOBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	16 OCTOBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	16 OCTOBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	16 OCTOBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	16 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	16 OCTOBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application	16 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	16 OCTOBER 2022
Technology Architecture	architecture diagram.	16 OCTOBER 2022

Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	22 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	24 OCTOBER 2022

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

7.1 Feature 1

Detection:

Obtaining these types of features requires active scan to target domain. Page contents are processed for us to detect whether target domain is used for phishing or not. Some processed information about pages are given below.

- Page Title
- Meta Tags
- Text in the Body
- Images etc.,

By analysing these information formation, we can gather information such as:

- Is it required to login to Wesite
- Website category
- information about audience profile etc.,

All of features explained above are useful for phishing domain detection. In some cases, it may not be useful to use some of these, so there are some limitations for using these features. For example, it may not be logical to use some of the features such as ContentBased Features for the developing fast detection mechanism which is able to analyze the number of domains between 100.000 and 200.000. Another example would be, if we want to analyze new registered domains Page-Based Features is not very useful. Therefore, the features that will be used by the detection mechanism depends on the purpose of the detection mechanism. Which features to use in the detection mechanism should be selected carefully.

CODE:

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <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,classifier,python">
[redacted]
    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+Nc
Pb1dKGj7Sk" crossorigin="anonymous">

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

</head>
[redacted]
<body>

<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>
[redacted]
</div>
[redacted]
    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
```

```
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+
OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.j
s"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox
MfooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min
.js"
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7B
h/kR0JKI"
        crossorigin="anonymous"></script>
```

[redacted]

```
    <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>
[redacted]
</html>
```

app.py

```
#importing required libraries
[redacted]
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
[redacted]
file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()
[redacted]
app = Flask(__name__)
[redacted]
@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) [redacted]
if __name__ == "__main__":
    app.run(debug=True)
```

features.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
[redacted]
class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""
[redacted]
        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass
[redacted]
        try:
            self.urlparse = urlparse(url)
            self.domain = self.urlparse.netloc
        except:
            pass
[redacted]
        try:
            self.whois_response = whois.whois(self.domain)
        except:
            pass
[redacted]

[redacted]
        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())

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

[redacted]

```
    # 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|i
s\.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|l
nkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.i
m|'

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

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

[redacted]

```
# 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
```

[redacted]

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

[redacted]

```
    # 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
```

[redacted]

```
            age = (expiration_date.year-creation_date.year)*12+
(expiration_date.month-creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1
[redacted]
    # 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
[redacted]
    # 11. NonStdPort
    def NonStdPort(self):
        try:
            port = self.domain.split(":")
            if len(port)>1:
          return -1
            return 1
        except:
            return -1
[redacted]
    # 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
```

[redacted]

```
            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
```

[redacted]

```
            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
```

[redacted]

```
            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
[redacted]
            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
[redacted]
        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
[redacted]
        except:
            return -1
[redacted]
    # 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
[redacted]
            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
[redacted]
            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
[redacted]
    # 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
[redacted]
    # 17. InfoEmail
    def InfoEmail(self):
```

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

[redacted]

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

[redacted]

```
# 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
```

[redacted]

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

[redacted]

```
            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
```

[redacted]

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

[redacted]

```
    # 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
```

[redacted]

```
    # 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 =
e.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.18
5\.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|17
5\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.23
2\.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|1
95\.16\.127\.102|195\.16\.127\.157|'
'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.6
4\.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\.8
6\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.23
1\.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

The experimental system is structured to empirically test and verify the efficacy of the proposed methods for phishing website detection. For training and evaluating the proposed techniques, three phishing datasets from the UCI repositories are used and the K-fold (where k = 10) cross-validation (CV) approach is used for the creation and evaluation of the phishing models.

The 10-fold CV option is based on its ability to create phishing models with the low impact of the issue of class imbalance [49, 50, 51, 52, 53]. Moreover, the K-fold CV approach ensures that each instance can be used iteratively for both training and testing [54, 55, 56].

On phishing datasets, based on 10-fold CV, the proposed methods and the chosen baseline classifiers (NB, SMO, SVM, and Decision Table (Dec Table)) are then implemented.

The phishing detection efficiency of the developed phishing models is then tested and contrasted with other experimented methods of phishing detection. All experiments were performed using the WEKA machine learning tool in the same environment [57].

8.2 User Acceptance Testing

Phishing attacks around the world cost billions of dollars in loss every year (Mc Combie et.al, 2009). Phishing has a huge negative impact on organizations' client relationships revenues, marketing pains and general corporate appearance (Dhanalakshmi et.al, 2011). Statistics report that 35.9% of financial sector is the target of Phishing frauds (APWG, 2010).

However these approaches are cannot detect and identify fresh phishes because of lists, where maintenance and human resources required and the scalability and run time are not suitable. This is the reason the list based approaches combine with other approaches [2, 3, 10,

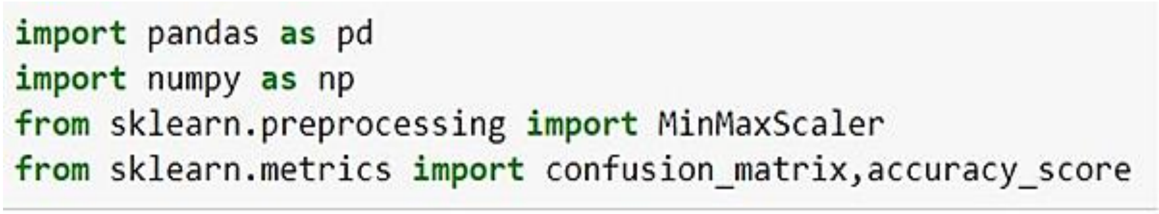
[12][13][14][15][16][17][18][19][20].

The Heuristics based approaches are predicted through one or more websites features like URL, source code and visual features.

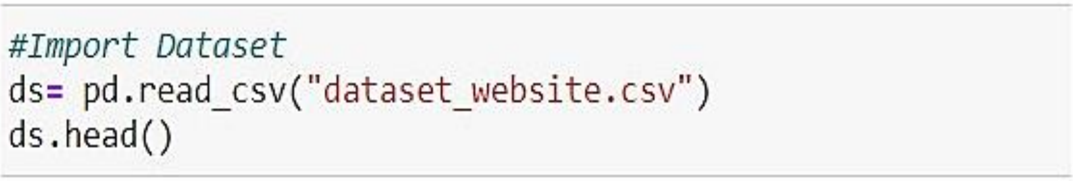
9. RESULTS

9.1 Performance Matrix

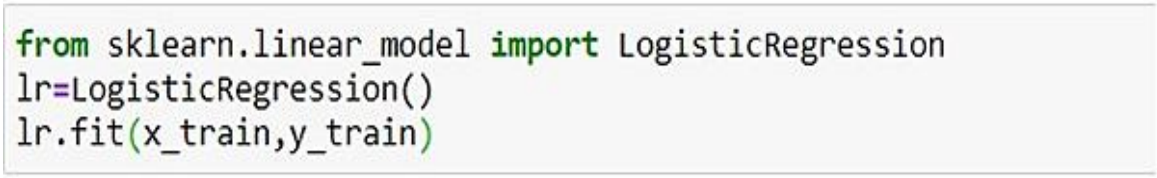
Coding For Metrics:



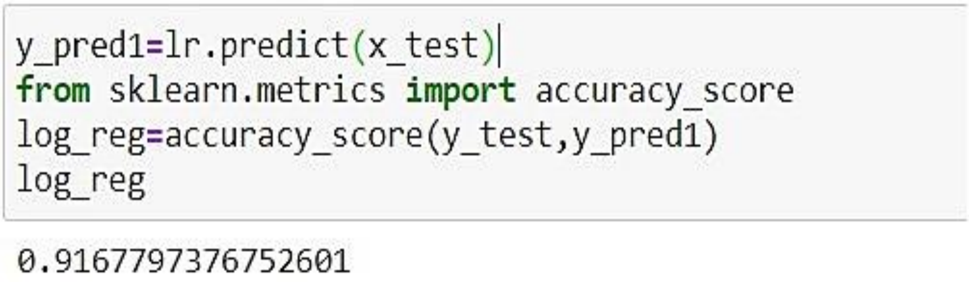
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
```



```
#Import Dataset
ds= pd.read_csv("dataset_website.csv")
ds.head()
```

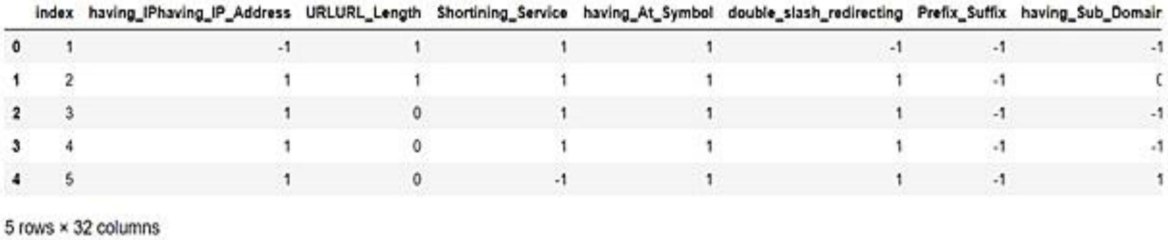


```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```



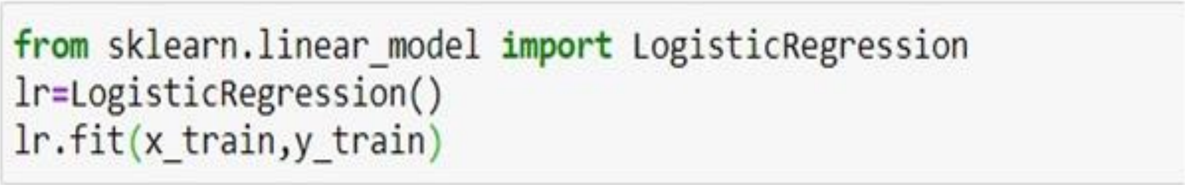
```
y_pred1=lr.predict(x_test)
from sklearn.metrics import accuracy_score
log_reg=accuracy_score(y_test,y_pred1)
log_reg
```

```
0.9167797376752601
```

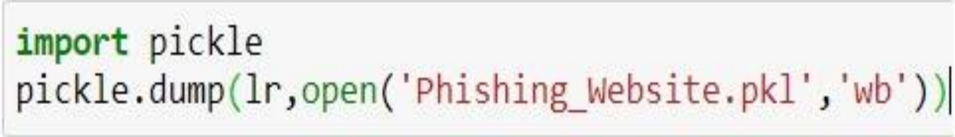


	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domair
0	1	-1	1	1	1	-1	-1	-1
1	2	1	1	1	1	1	-1	0
2	3	1	0	1	1	1	-1	-1
3	4	1	0	1	1	1	-1	-1
4	5	1	0	-1	1	1	-1	1

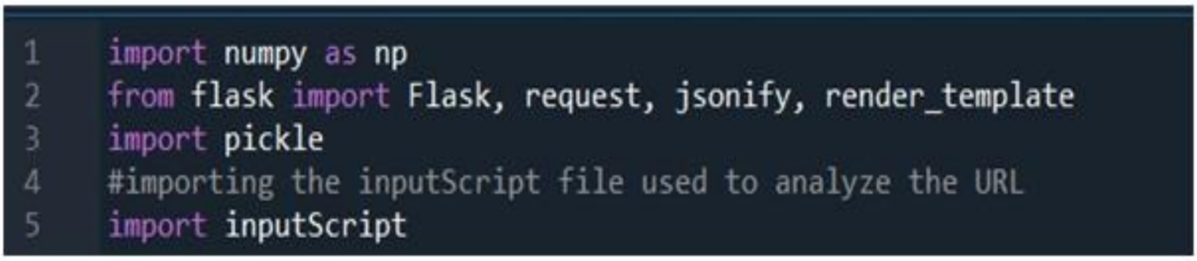
5 rows × 32 columns



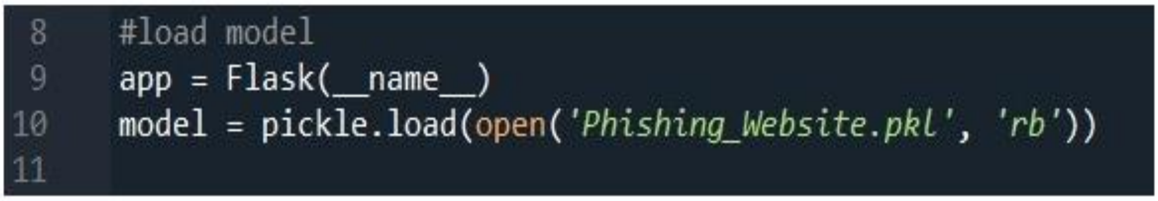
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```



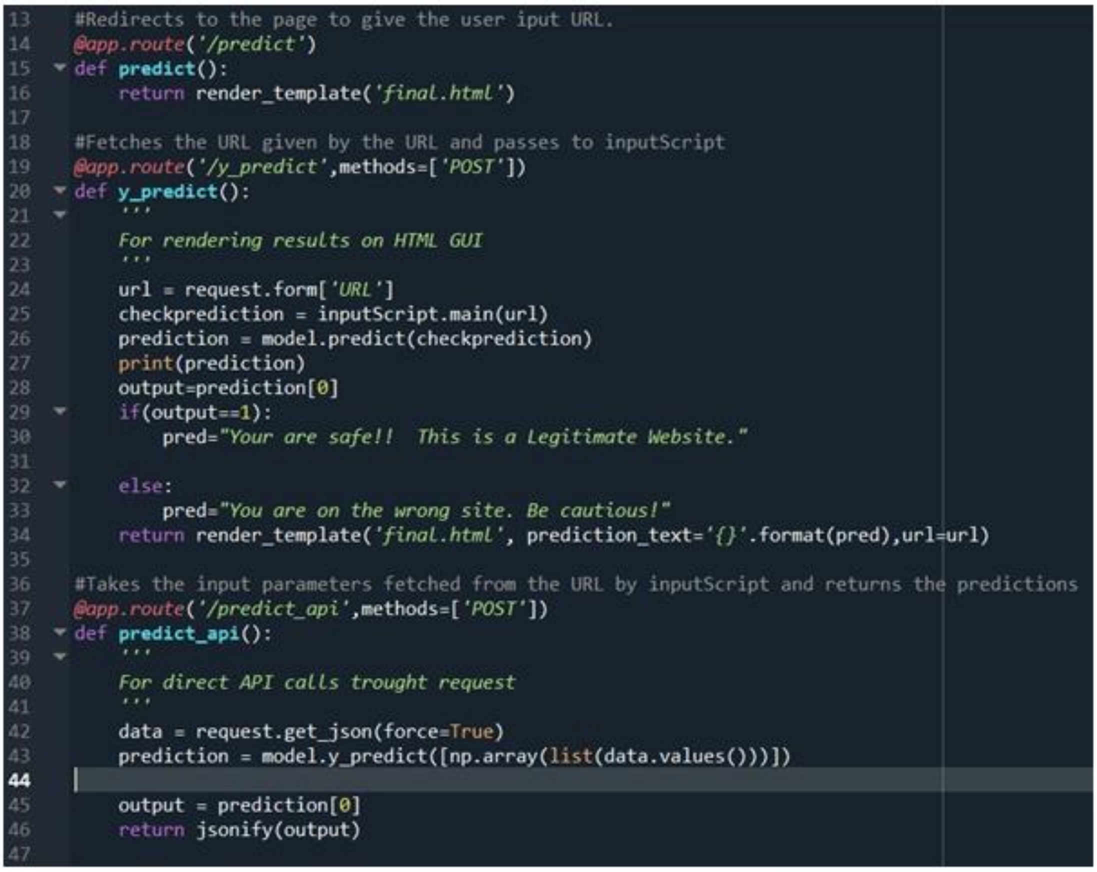
```
import pickle
pickle.dump(lr,open('Phishing_Website.pkl','wb'))
```



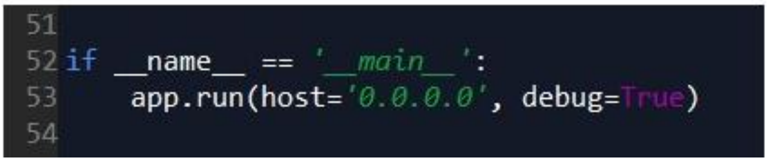
```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
#importing the inputScript file used to analyze the URL
import inputScript
```



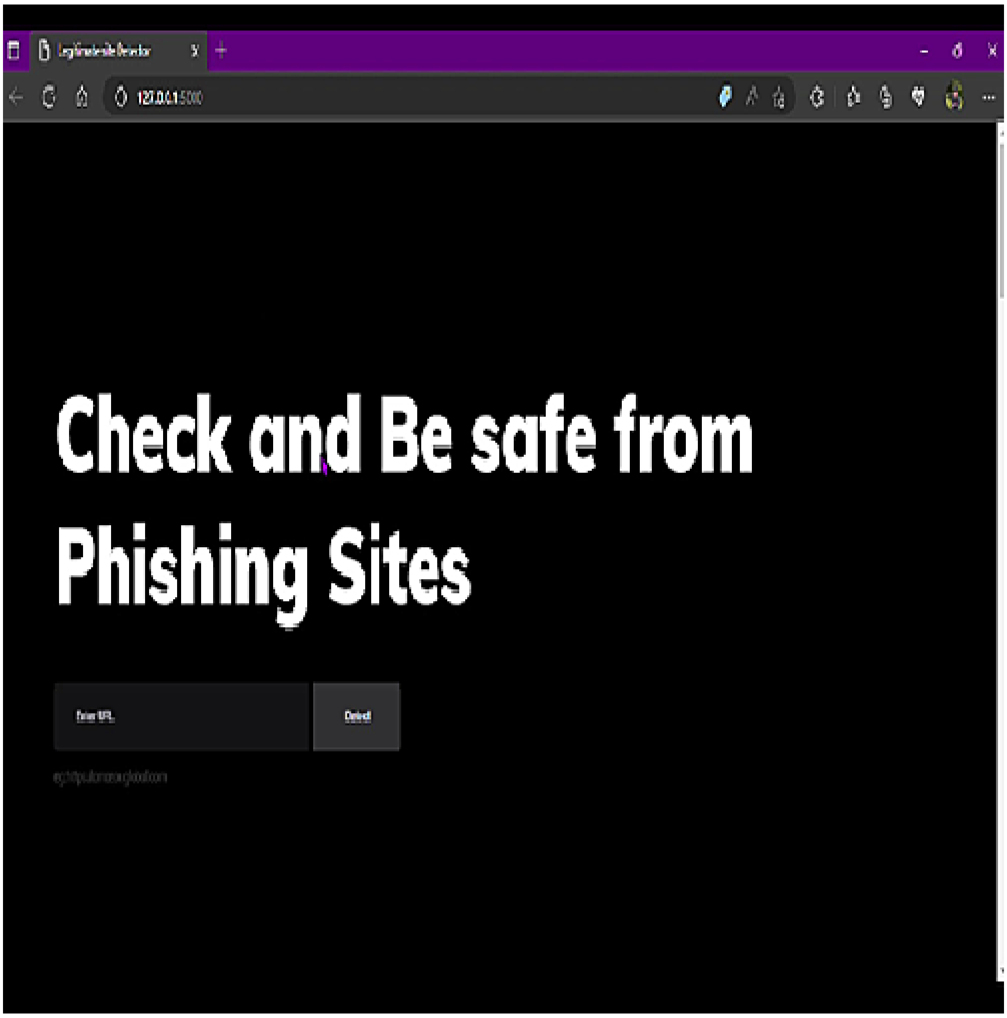
```
#load model
app = Flask(__name__)
model = pickle.load(open('Phishing_Website.pkl', 'rb'))
```



```python
#Redirects to the page to give the user iput URL.
@app.route('/predict')
def predict():
    return render_template('final.html')

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    url = request.form['URL']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="Your are safe!!  This is a Legitimate Website."

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('final.html', prediction_text='{}'.format(pred),url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

```



```python

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```



Check and Be safe from Phishing Sites

10 .Advantages And Disadvantages

Advantages:

1. Could Easily find the phishing websites before clicking on them

2. Can Identify Which Sites Have The Most Target

3. Could save your data before it is too late

4. DETECTION is Better Than Cure.

Disadvantages:

5. Hackers Find New Way To Attack,

6. May not be able to detect all the websites.

7. Could possibly bypass the detection.

11.Conclusion

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users.

Unfortunately, many of the existing phishing-detection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm that is often caused by either a delay in blacklist update as a result of human verification process involved in classification or perhaps, it can be attributed to human error in classification which may lead to improper classification of the classes.

These critical issues have drawn many researchers to work on various approaches to improve detection accuracy of phishing attacks and to minimize false alarm rate. The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model.

Therefore, it requires an effective technique to regulate retraining as to enable machine learning algorithm to actively adapt to the changes in phish patterns.

12.Future Scope

Despite there are several ways to carry out these attacks, unfortunately the current phishing detection techniques cover some attack vectors like email and fake websites. Therefore, building a specific limited scope detection system will not provide complete protection from the wide phishing attack vectors.

13.Source Code

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <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,classifier,python">
[redacted]
    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.
min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+Nc
Pb1dKGj7Sk" crossorigin="anonymous">

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

</head>
```

[redacted]

```
<body>

<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>
[redacted]
</div>
[redacted]
    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+
OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.j
s"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox
MfooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min
.js"
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7B
h/kR0JKI"
        crossorigin="anonymous"></script>
[redacted]

    <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>
```

[redacted]

```
</html>
```

TEAM GITHUB LINK: https://github.com/IBM-EPBL/IBM-Project-5743-1658813865

Demo Video :