WEB PHISHING DETECTION

IBM-Project-39584-1660462497

WEB PHISHING DETECTION APPLICATION

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

Submitted by

BANUPRIYA M (950819104009)
DEEPIKA K (950819104015)
SANGEETHA M (950819104039)
SUTHA K R (950819104046)

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

Government College of Engineering
TIRUNELVELI– 627007

INDEX

1.INTRODUCTION

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. Phishing attacks can paralyze a business. Staff might be unable to continue their work. Data and assets might be stolen or damaged. Customers might be unable to access online services. The reason security defenders struggle to detect phishing domains is because of the unique part of the website domain.

1.1 Project Overview

Category: Machine Learning

Team ID : PNT2022TMID33766

💻Skills Required:

Python,Python Web Frame Works,Python For Data Visualization,Data Preprocessing Techniques,Machine Learning,IBM Cloud,IBM Watson Studio,Python-Flask

Project Description:
Phishing is a form of fraudulent attack where the attacker tries to gain sensitive information by posing as a reputable source. In a typical phishing attack, a victim opens a compromised link that poses as a credible website. The victim is then asked to enter their credentials, but since it is a “fake”

website, the sensitive information is routed to the hacker and the victim gets "'hacked."
Phishing is popular since it is a low effort, high reward attack. Most modern web browsers, antivirus software and email clients are pretty good at detecting phishing websites at the source, helping to prevent attacks. To understand how they work, this project shows you how to build your own phishing URL detector using Python and Applied data science:

1.Identify the criteria that can recognize fake URLs
2.Build a decision tree that can iterate through the criteria
3.Train our model to recognize fake vs real URLs
4,Evaluate our model to see how it performs
5.Check for false positives/negatives

Social Impact:
• It will help to minimize the frauds while using software solutions(EX: Web applications, etc).

Business Model/Impact:
• This application can be used by many E-commerce enterprises in order to make the whole transaction process secure.

1.2 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.

2. LITERATURE SURVEY

2.1 Existing problem

Link : https://checkphish.ai/

2.2 References

[1] JIAN MAO1,WENQIAN TIAN1, PEI LI1, TAO WEI2, AND ZHENKAI LIANG3 Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity.

[2] Zou Futai, Gang Yuxiang, Pei Bei, Pan Li, Li Linsen Web Phishing Detection Based on Graph Mining.

[3] Nick Williams, Shujun Li Simulating human detection of phishing websites: An investigation into the applicability of ACT-R cognitive behaviour architecture model.

[4] XIN MEI CHOO, KANG LENG CHIEW,DAYANG HANANI ABANG IBRAHIM,NADIANATRA MUSA, SAN NAH SZE, WEI KING TIONG Feature-based Phishing Detection Technique.

[5] Giovanni Armano, Samuel Marchal and N. Asokan RealTime Client-Side Phishing Prevention Add-on.

[6] Trupti A. Kumbhare and Prof. Santosh V. Chobe An Overview of Association Rule Mining Algorithms.

[7] S.Neelamegam, Dr.E.Ramaraj Classification algorithm in Data mining: An Overview

[8] Varsharani Ramdas Hawanna, V. Y. Kulkarni and R. A. Rane A Novel Algorithm to Detect Phishing URLs.

[9] Jun Hu, Xiangzhu Zhang,Yuchun Ji, Hanbing Yan, Li Ding, Jia Li and Huiming Meng Detecting Phishing Websites Based on the Study of the Financial Industry Webserver Logs. [10] Samuel Marchal, Giovanni Armano and Nidhi Singh Offthe-Hook: An Efficient and Usable.

[10] Samuel Marchal, Giovanni Armano and Nidhi Singh Offthe-Hook: An Efficient and Usable.

[11] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. “Machine Learning-Based Phishing Detection from URLs,” Expert Systems with Applications, vol. 117, pp. 345-357, January 2019.

[12] J. James, Sandhya L. and C. Thomas, “Detection of phishing URLs using machine learning techniques,” International Conference on Control Communication and Computing (ICCC),
December 2013.

[13] Pradeepthi, K. V., & Kannan, A. “Performance study of classification techniques for phishing URL detection,” Sixth International Conference on Advanced Computing (IcoAC), December 2014.

[14] Dipayan Sinha, Dr. Minal Moharir, Prof. Anitha Sandeep, “Phishing Website URL Detection using Machine Learning,” International Journal of Advanced Science and Technology, vol. 29, no. 3, pp. 2495-2504, 2020.

[15] R. Kiruthiga, D. Akila, “Phishing Websites Detection Using Machine Learning,” International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 2S11, pp. 11-114, September 2019

2.3 Problem Statement Definition

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.

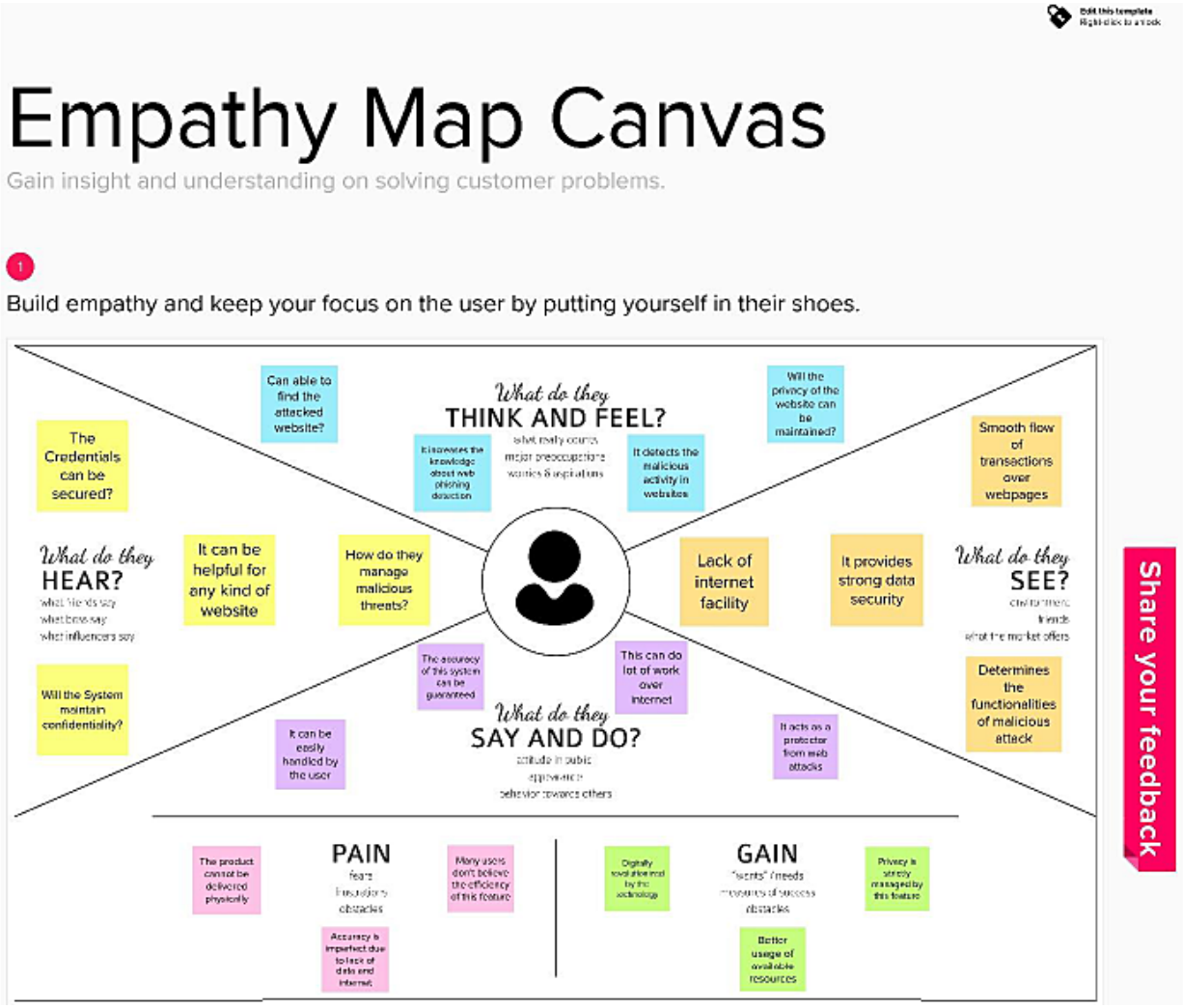
Reference: https://miro.com/templates/customer-problem-statement/

Example:Web Phishing Detection

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A social influencer	Access the original webpage	I am unable to access the original webpage	url redirect to the cloned webpage	discomfort
PS-2	Student	Update my aadhar	I am unable to access the original webpage	url redirect to the cloned web page	insecure

3. IDEATION & PROPOSED SOLUTION

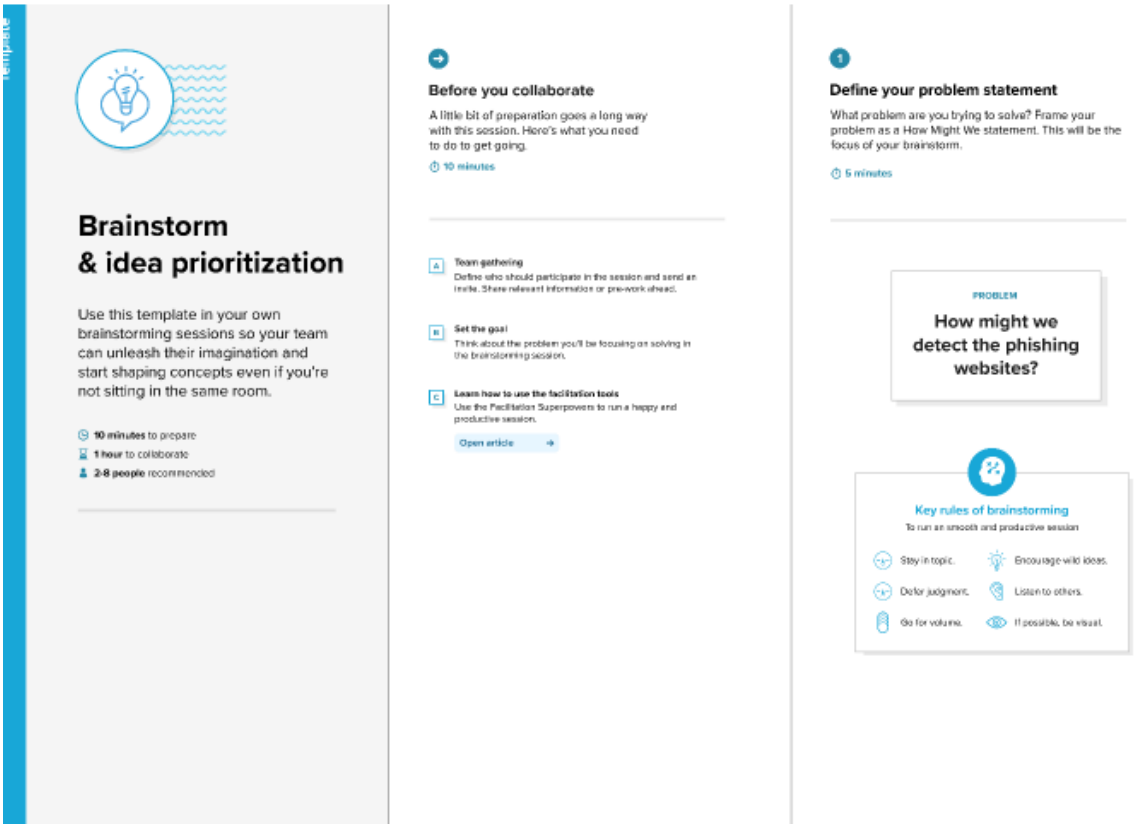
3.1 Empathy Map Canvas



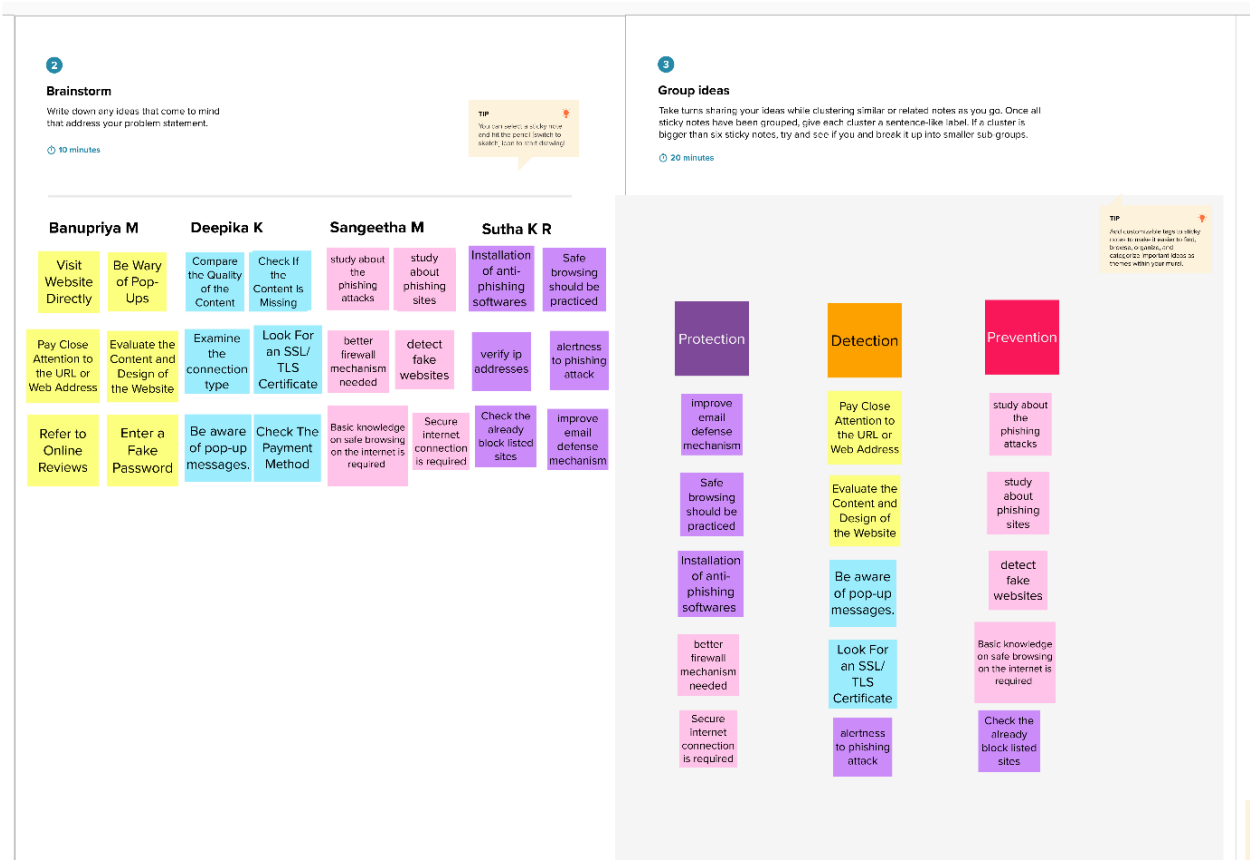
3.2 Ideation & Brainstorming

Web Phishing Detection:

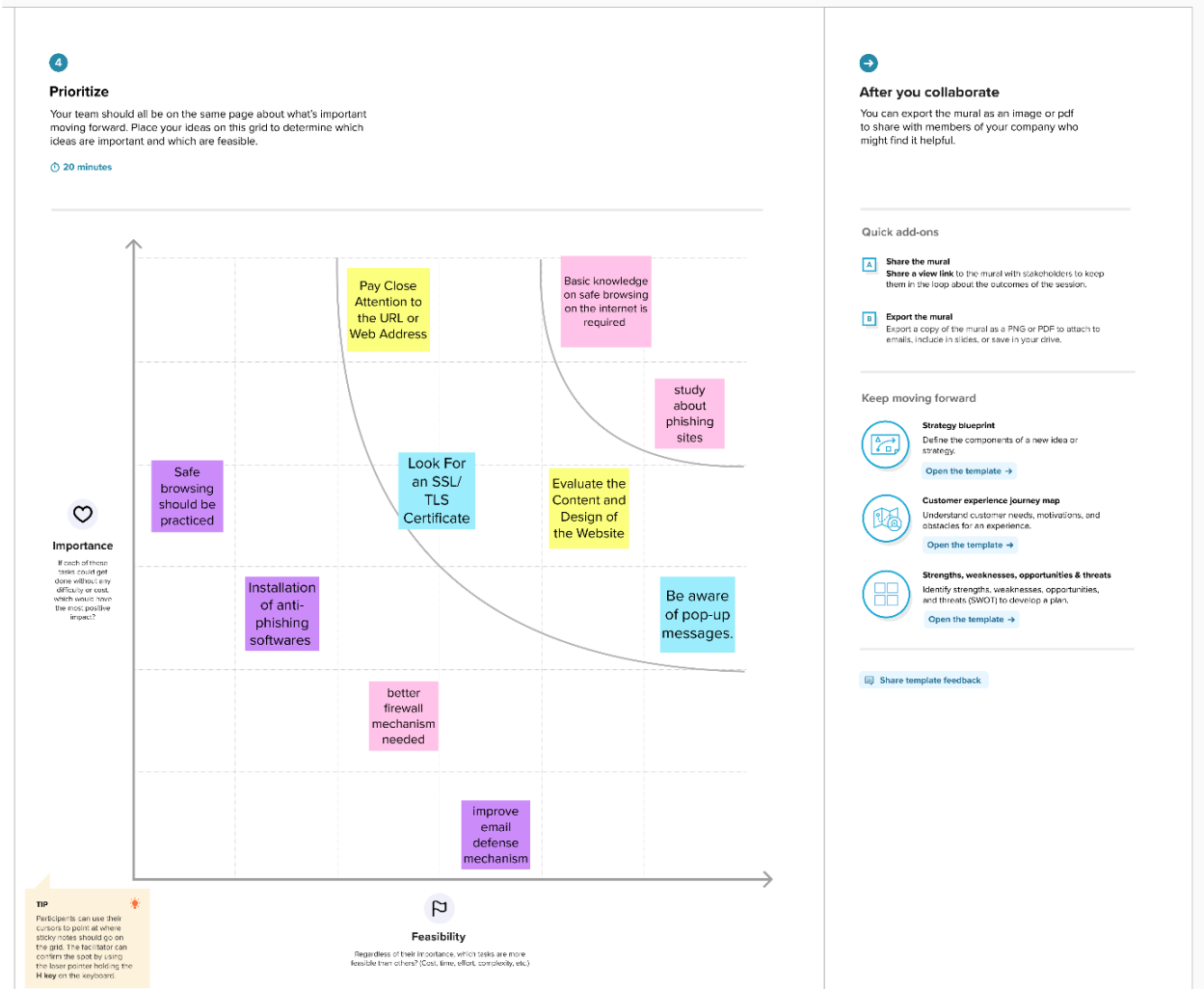
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and

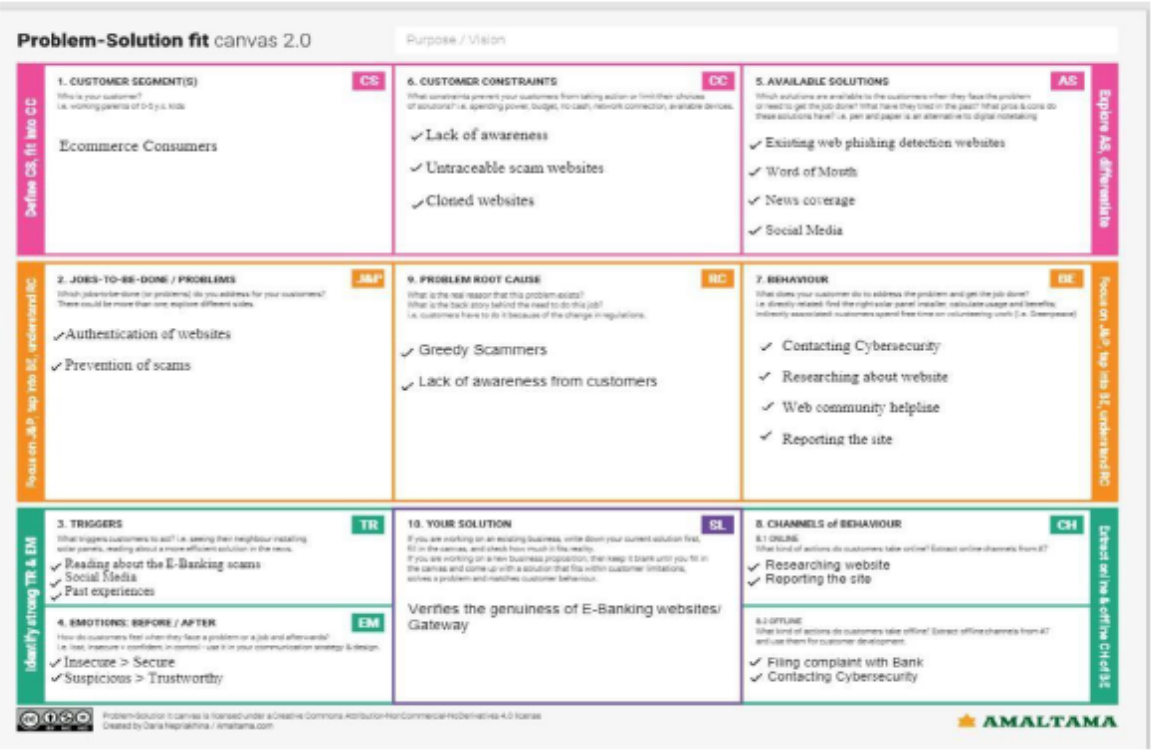
		techniques to extract the phishing datasets criteria to classify their legitimacy.
3.	Novelty / Uniqueness	The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website ornot.

4.	Social Impact / Customer Satisfaction	The feasibility of implementing this idea is moderate neither easy nor tough because the system needs to satisfy the basic requirements of the customer as well as it should act as a bridge towards achieving high accuracy on predicting and analyzing the detected websites or files to protect our customer to the fullest.

5.	Business Model (Revenue Model)	People buy subscription annually, to protect their files both locally and at remote location with the help of our cloud integrated flask app for web phishing detection.

3.4 Problem Solution fit

Project Title: Web Phishing Detection **Project Design Phase-I - Solution Fit Template** **Team ID: PNT2022TMID33766**



4. REQUIREMENT ANALYSIS

4.1 Functional requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	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

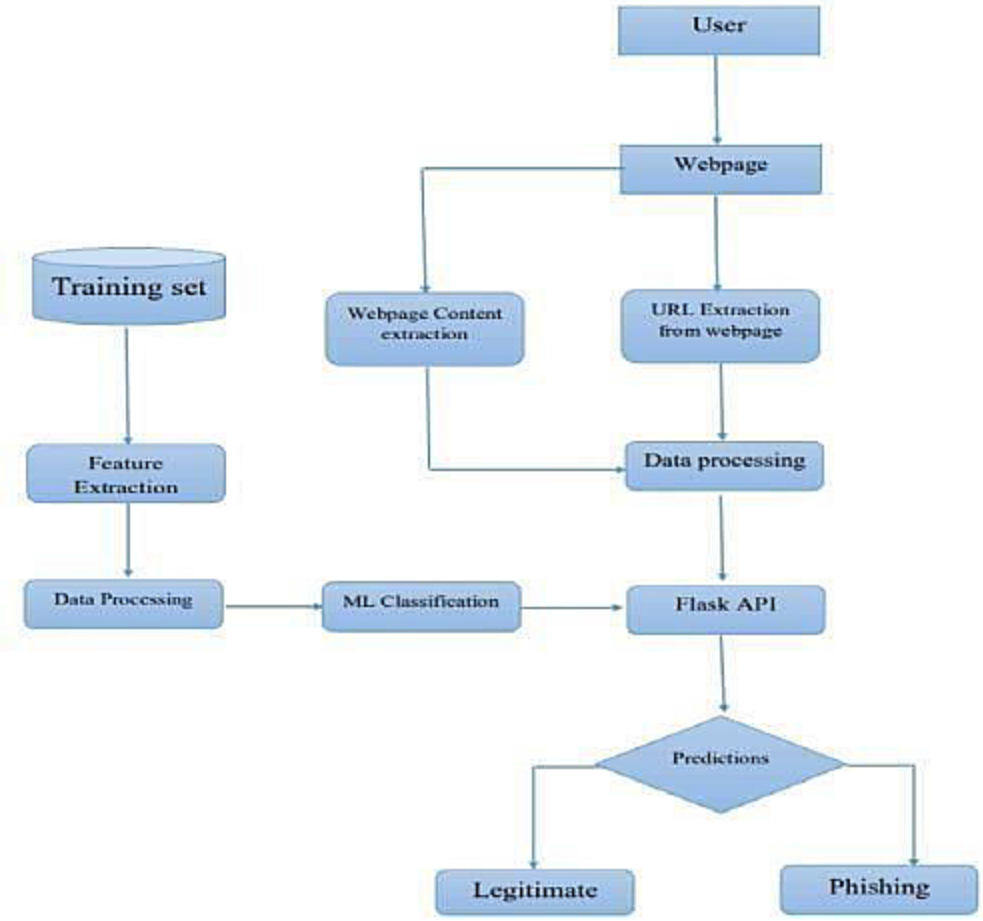
NFR-1	**Usability**	Usability is commonly considered to be the enemy of security. In general, being secure means taking extra steps to avoid falling for different attacks. This is especially true of phishing where the best ways to prevent against most phishing attacks are commonly known, but cyber security guidance is rarely followed.
NFR-2	**Security**	Phishing is a type of cyber security attack during which malicious actors send messages pretending to be a trusted person or entity. Lack of security awareness among employees is also one of the major reasons for the success of phishing.

NFR-3	**Reliability**	Reliability Factor is determined on the basis of the outcome of these strata, using Rough Set Theory . Reliability Factor determines the possibility of a suspected site to be Valid or Fake. Using Rough set theory most and the least influential factors towards phishing are also determined.
NFR-4	**Performance**	The two main characteristics of a phishing site are that it looks extremely similar to a legitimate site and that it has at least one field to enable users to input their credentials. A common indicator of a phishing attempt is a suspicious attachment.

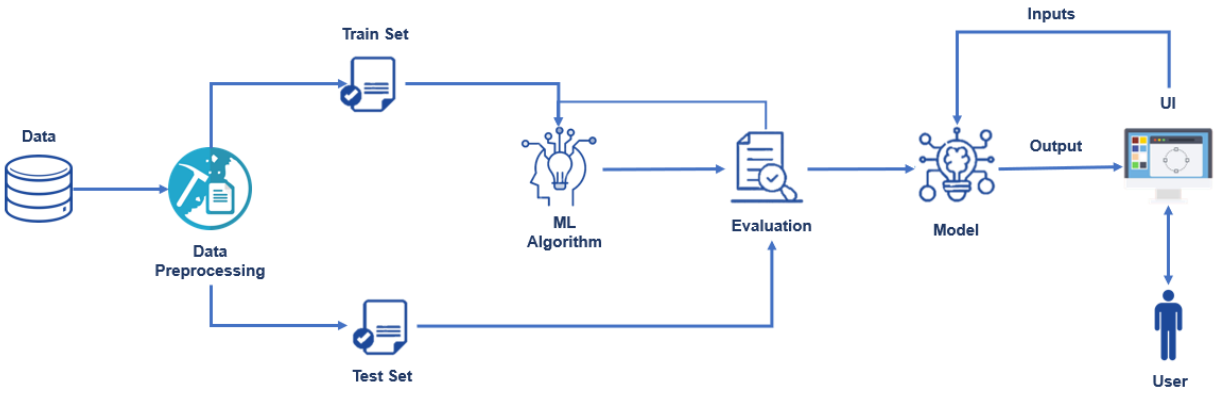
NFR-5	**Availability**	Phishing is a type of social engineering attack often used to steal user data, including login credentials and credit card numbers. It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.
NFR-6	**Scalability**	Scalable detection and isolation of phishing,the main ideas are to move the protection from end users towards the network provider and to employ the novel bad neighbourhood concept, in order to detect and isolate both phishing email senders and phishing web servers.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

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 (Web user)	User input	USN-1	As a user I can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After I compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a user I can have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN.	In this I can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here I will send all the model output to classifier in order to produce final result	In this I will find the correct classifier for producing the result	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and 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	2	High	BANUPRIYA M
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	High	DEEPIKA K
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extracts feature using heuristic and visual similarity.	2	Low	SANGEETHA M
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	2	Medium	SUTHA K R
Sprint-3	Classifier	USN-5	Model then displays whether the website is legal site or a phishing site	1	High	BANUPRIYA M
Sprint-3	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site	1	High	DEEPIKA K
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	SUTHA K R

Project Tracker, Velocity & Burndown Chart:

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

Velocity:

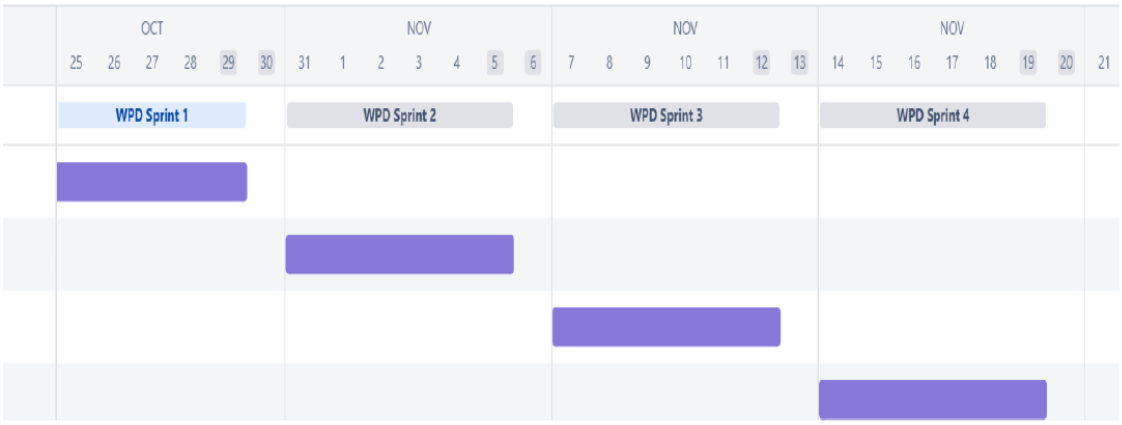
We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). So our team's average velocity (AV) periteration unit (story points per day)

AV = (Sprint Duration / Velocity)

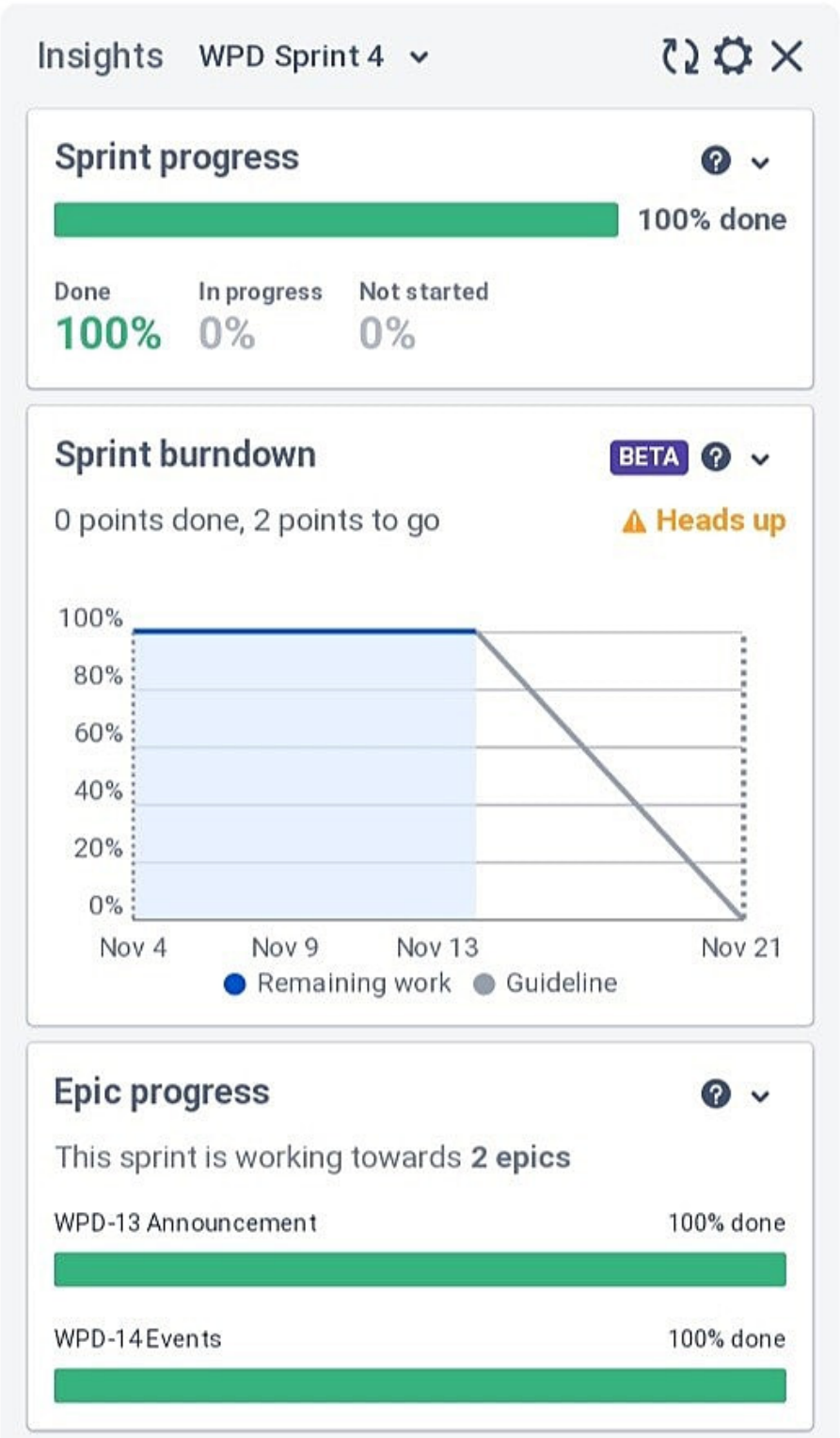
= 20 /10

AV = 2

6.2 Sprint Delivery Schedule



6.3 Reports from JIRA



7. CODING & SOLUTIONING

app.py:

```
import numpy as np
import pandas
from flask import Flask, request, jsonify, render_template
import pickle
import inputScript
app = Flask(__name__)
model = pickle.load(open('Phishing_Website.pkl','rb'))
@app.route('/')
def home():
    return render_template('index.html')
ans = ""
bns = ""
@app.route('/y_predict', methods=['POST','GET'])
def y_predict():
    url = request.form['url']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="You are safe!!  This is a legitimate Website."
```

```
        return render_template('index.html',bns=pred)


    else:
        pred="You are on the wrong site. Be cautious!"
        return render_template('index.html',ans=pred)
@app.route('/predict_api', methods=['POST'])
def predict_api():
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])
    output=prediction[0]
    return jsonify(output)
if __name__ == '__main__':
    app.run()
# In[10]:
# In[ ]:
'''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'))
@app.route('/')
```

```
# def helloworld():
#     return render_template("index.html")
#Redirects to the page to give the user input URL.
@app.route('/predict')
def predict():
    return render_template('index.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.FeatureExtraction(url)
    print(checkprediction)
    prediction = model.predict(np.array(checkprediction.features).reshape(-
1,30))
    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('index.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)
if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
    '''
```

inputScript.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
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse
import favicon
import regex
```

```
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import datetime
import requests
import re
"""
Check if URL contains any IP address. Returns -1 if contains else returns 1
"""
def having_IPhaving_IP_Address(url):
    match=regex.search(
  '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|' #IPv4
                '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'  #IPv4 in hexadecimal
                '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)
   #Ipv6
    if match:
       #print match.group()
       return -1
    else:
       #print 'No matching pattern found'
```

```
        return 1
"""
Check for the URL length. Return 1 (Legitimate) if the URL length is less
than 54 characters
Return 0 if the length is between 54 and 75
Else return -1
"""
def URLURL_Length (url):
    length=len(url)
    if(length<=75):
        if(length<54):
            return 1
        else:
            return 0
    else:
        return -1

"""
Check with the shortened URLs.
Return -1 if any shortened URLs used.
Else return 1
"""
def Shortining_Service (url):

match=regex.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyu
```

```
rl|tr\.im|is\.gd|cli\.gs|'

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

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

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

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

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


#Checking for @ symbol. Returns 1 if no @ symbol found. Else returns 0.
def having_At_Symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
```

```
        return 1
    else:
        return -1


#Checking for Double Slash redirections. Returns -1 if // found. Else returns
1
def double_slash_redirecting(url):
    for i in range(8,len(url)):
        if(url[i]=='/'):


            if(url[i-1]=='/'):
                return -1
    return 1


#Checking for - in Domain. Returns -1 if '-' is found else returns 1.
def Prefix_Suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return -1
    else:
        return  1


"""
Check the Subdomain. Return 1 if the subDomain contains less than 1 '.'
```

```
Return 0 if the subDomain contains less than 2 '.'
Return -1 if the subDomain contains more than 2 '.'
"""
def having_Sub_Domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')<=2):
        if(subDomain.count('.')<=1):
            return 1
        else:
            return 0
    else:
        return -1

#Checking the SSL. Returns 1 if it returns the respomse code and -1 if exceptions are thrown.
def SSLfinal_State(url):
    try:
        response = requests.get(url)
        return 1
    except Exception as e:
        return -1

#domains expires on ≤ 1 year returns -1, otherwise returns 1
```

```
def Domain_registeration_length(url):
    try:
        domain = whois.whois(url)
        exp=domain.expiration_date[0]
        up=domain.updated_date[0]
        domainlen=(exp-up).days
        if(domainlen<=365):
            return -1
        else:
            return 1
    except:
        return -1


#Checking the Favicon. Returns 1 if the domain of the favicon image and the URL domain match else returns -1.
def Favicon(url):
    subDomain, domain, suffix = extract(url)
    b=domain
    try:
        icons = favicon.get(url)
        icon = icons[0]
        subDomain, domain, suffix =extract(icon.url)
        a=domain
        if(a==b):
```

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


#Checking the Port of the URL. Returns 1 if the port is available else returns -1.
def port(url):
    try:
        a_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        location=(url[7:],80)
        result_of_check = a_socket.connect_ex(location)
        if result_of_check == 0:
            return 1
        else:
            return -1
        a_socket.close
    except:
        return -1


# HTTPS token in part of domain of URL returns -1, otherwise returns 1
def HTTPS_token(url):
    match=re.search('https://|http://',url)
```

```
    if (match and match.start(0)==0):
        url=url[match.end(0):]
    match=re.search('http|https',url)
    if match:
        return -1
    else:
        return 1

#% of request URL<22% returns 1, otherwise returns -1
def Request_URL(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
```

```
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return 1
        else:
            return -1
    except:
        return -1

#:% of URL of anchor<31% returns 1, % of URL of anchor ≥ 31% and ≤
67% returns 0, otherwise returns -1
def URL_of_Anchor(url):
```

```
try:
    subDomain, domain, suffix = extract(url)
    websiteDomain = domain


    opener = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(opener, 'lxml')
    anchors = soup.findAll('a', href=True)
    total = len(anchors)
    linked_to_same = 0
    avg = 0
    for anchor in anchors:
        subDomain, domain, suffix = extract(anchor['href'])
        anchorDomain = domain
        if(websiteDomain==anchorDomain or anchorDomain==''):
            linked_to_same = linked_to_same + 1
    linked_outside = total-linked_to_same
    if(total!=0):
        avg = linked_outside/total


    if(avg<0.31):
        return 1
    elif(0.31<=avg<=0.67):
        return 0
    else:
```

```
            return -1
    except:
        return 0


"""
% of links in <meta>, <script>and<link>tags < 25% returns 1, % of links in <meta>,
<script> and <link> tags ≥ 25% and ≤ 81% returns 0, otherwise returns -1
"""


def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
```

```
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

#Server Form Handling
#SFH is "about: blank" or empty → phishing, SFH refers to a different domain → suspicious, otherwise → legitimate
def SFH(url):
    #ongoing
    return -1
```

```
#:using "mail()" or "mailto:" returning -1, otherwise returns 1
def Submitting_to_email(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:','mail():')):
            return -1
        else:
            return 1
    except:
        return -1

#Host name is not in URL returns -1, otherwise returns 1
def Abnormal_URL(url):
    subDomain, domain, suffix = extract(url)
    try:
        domain = whois.whois(url)
        hostname=domain.domain_name[0].lower()
        match=re.search(hostname,url)
        if match:
            return 1
        else:
            return -1
```

```
    except:
        return -1

#number of redirect page ≤ 1 returns 1, otherwise returns 0
def Redirect(url):
    try:
        request = requests.get(url)
        a=request.history
        if(len(a)<=1):
            return 1
        else:
            return 0

    except:
        return 0

#onMouseOver changes status bar returns -1, otherwise returns 1
def on_mouseover(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_script =0
        for meta in soup.find_all(onmouseover=True):
```

```
            no_of_script = no_of_script+1
        if(no_of_script==0):
            return 1
        else:
            return -1
    except:
        return -1

#right click disabled returns -1, otherwise returns 1
def RightClick(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find_all('script',mousedown=True)):
            return -1
        else:
            return 1
    except:
        return -1

#popup window contains text field → phishing, otherwise → legitimate
def popUpWidnow(url):
    #ongoing
    return 1
```

```
#using iframe returns -1, otherwise returns 1
def Iframe(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        nmeta=0
        for meta in soup.findAll('iframe',src=True):
            nmeta= nmeta+1
        if(nmeta!=0):
            return -1
        else:
            return 1
    except:
        return -1

#:age of domain ≥ 6 months returns 1, otherwise returns -1
def age_of_domain(url):
    try:
        w = whois.whois(url).creation_date[0].year
        if(w<=2018):
            return 1
        else:
            return -1
```

```
    except Exception as e:
        return -1


#no DNS record for domain returns -1, otherwise returns 1
def DNSRecord(url):
    subDomain, domain, suffix = extract(url)
    try:
        dns = 0
        domain_name = whois.whois(url)
    except:
        dns = 1


    if(dns == 1):
        return -1
    else:
        return 1


#website rank < 100.000 returns 1, website rank > 100.000 returns 0,
otherwise returns -1
def web_traffic(url):
    try:
        rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat
=s&url=" + url).read(), "lxml").find("REACH")['RANK']
    except TypeError:
```

```
        return -1
    rank= int(rank)
    if (rank<100000):
        return 1
    else:
        return 0


#:PageRank < 0,2 → phishing, otherwise → legitimate
def Page_Rank(url):
    #ongoing
    return 1

#webpage indexed by Google returns 1, otherwise returns -1
def Google_Index(url):
    try:
        subDomain, domain, suffix = extract(url)
        a=domain + '.' + suffix
        query = url
        for j in search(query, tld="co.in", num=5, stop=5, pause=2):
            subDomain, domain, suffix = extract(j)
            b=domain + '.' + suffix
        if(a==b):
            return 1
        else:
```

```
            return -1
    except:
        return -1
#:number of links pointing to webpage = 0 returns 1, number of links pointing to webpage> 0
#and ≤ 2 returns 0, otherwise returns -1

def Links_pointing_to_page (url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        count = 0
        for link in soup.find_all('a'):
            count += 1
        if(count>=2):
            return 1
        else:
            return 0
    except:
        return -1

#:host in top 10 phishing IPs or domains returns -1, otherwise returns 1
def Statistical_report (url):
    hostname = url
```

```
    h = [(x.start(0), x.end(0)) for x in
regex.finditer('https://|http://|www.|https://www.|http://www.', hostname)]
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in regex.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:
            hostname = hostname[:h[0][0]]

url_match=regex.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|e
sy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly',url)
    try:
        ip_address = socket.gethostbyname(hostname)

ip_match=regex.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\.15
1\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.22
5|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\.2
15\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.2
4\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131
|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\
.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.5
6\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|2
09\.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)
    except:
        return -1


    if url_match:
        return -1
    else:
        return 1


#returning scrapped data to calling function in app.py
def main(url):
    check = [[having_IPhaving_IP_Address
(url),URLURL_Length(url),Shortining_Service(url),having_At_Symbol(url),


double_slash_redirecting(url),Prefix_Suffix(url),having_Sub_Domain(url),S
SLfinal_State(url),


Domain_registeration_length(url),Favicon(url),port(url),HTTPS_token(url),R
equest_URL(url),


URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),A
bnormal_URL(url),


Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(u
rl),
```

```
age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Googl
e_Index(url),
            Links_pointing_to_page(url),Statistical_report(url)]]
    print(check)
    return check
```

index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Phishing Website Detection</title>
    <style>
    body {
        background-color: #92a8d1;
    }
    </style>
  </head>
  <body>
    <center>
      <h2><b>Phishing Website Detection</b></h2>
      <form name="form" action="/y_predict" method="post" class="body">
        <input
          type="text"
          id="url"
```

```
          name="url"
          placeholder="Enter a URL"
          size="50"
        />
        <button type="submit">Submit</button>
      </form>
      <h3 style="text-align: center; color: red; font-size: 20px">{{ans}}</h3>
      <h3 style="text-align: center; color: green; font-size: 20px">{{bns}}</h3>
    </center>
  </body>
</html>
```

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	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://phishingshield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		Banupriya M
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in 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://phishingshield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		Deepika K
LoginPage_TC_OO3	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://phishingshield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		Sangeetha M
LoginPage_TC_OO4	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://phishingshield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Sutha K R
LoginPage_TC_OO5	Functional	Home page	Testing the website with multiple URLs		1.Enter URL (https://phishingshield.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. google.com 2. onpricei.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Banupriya M

8.2 User Acceptance Testing

UAT Execution & Report Submission

1. Purpose of Document

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

2. Defect Analysis

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

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

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3

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

9.RESULTS

9.1 Performance Metrics

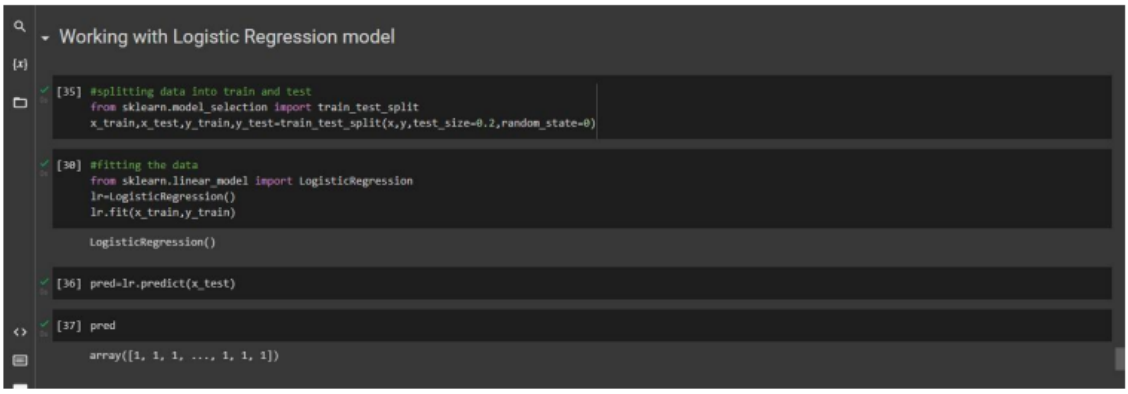
S.No.	Parameter	Values	Screenshot

1.	Metrics	**Regression Model: Logistic Regression**
		MAE – 0.26142017186793304
		MSE - 0.5228403437358661
		RMSE - 0.7230769971004928
		R2 score - -2.888673182487615
		Accuracy: 91.6%
		
		Classification Model: Decision Tree Classifier
		Confusion Matrix -
		array([[61, 249],
		[26, 1875]])
		Accuracy Score-
		0.8756218905472637
		Classification Report – refer screenshot
			`y_pred1 = lr.predict(x_test)`
			`from sklearn.metrics import accuracy_score`
			`log_reg = accuracy_score(y_test,y_pred1)`
			`log_reg`
			
			`0.9167797376752601`

2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Attached Below

1. METRICS:

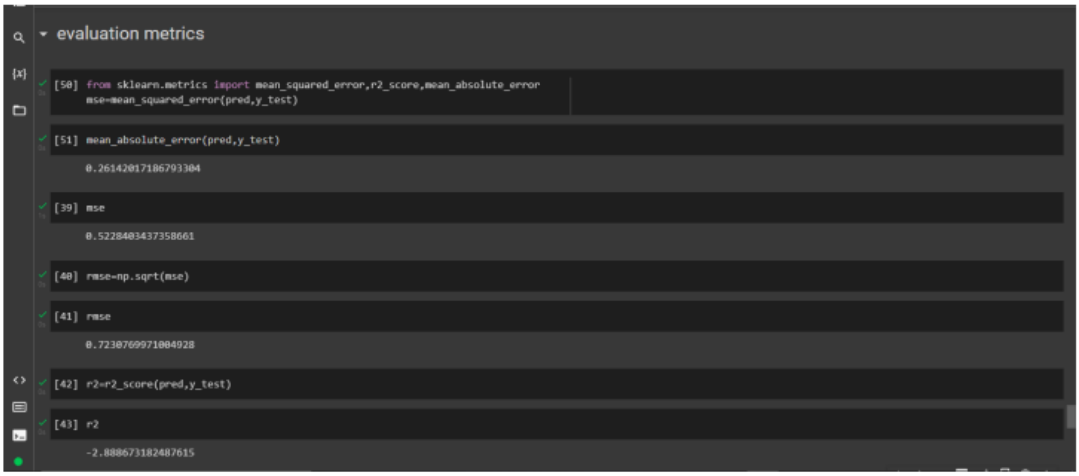
REGRESSION MODEL: LOGISTIC REGRESSION



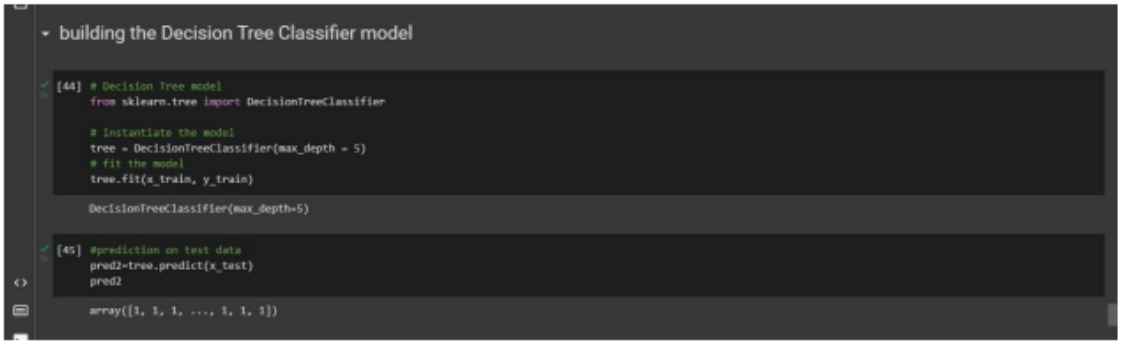
EVALUATION METRICS:

Here are some evaluation metrics used for regression they are,

- R2 Score
- Mean Square Error(MSE)
- RMSE(Root Mean Square Error)
- Mean Absolute Error(MAE)



CLASSIFICATION MODEL: DECISION TREE CLASSIFIER



EVALUATION METRICS:

Some of the evaluation metrics is as follows

- Confusion matrix
- Accuracy score
- Classification report

2. TUNE THE MODEL: DECISION TREE CLASSIFIER

HYPERPARAMETER TUNING:

10. ADVANTAGES & DISADVANTAGES

Advantages:

Blacklists:

- Requiring low resources on host machine
- Effective when minimal FP rates are required.

Heuristics and visual similarity:

- Mitigate zerohour attacks.

Machine Learning:

- Mitigate zerohour attacks.
- Construct own classification models.

Disadvantages:

- Mitigation of zero-hour phishing attacks.
- Can result in excessive queries with heavily loaded servers
- Higher FP rate than blacklists
- High computational cost.
- Time consuming.
- Costly.
- Huge number of rules.

11. CONCLUSION

Education awareness is the most significant strategy to protect users from phishing attacks. Internet users should be aware of all security recommendations made by professionals. Every user should also be taught not to mindlessly follow links to websites where sensitive information must be entered. Before visiting a website, make sure to check the URL. In the future, the system could be upgraded to automatically detect the web page and the application's compatibility with the web browser. Additional work can be done to distinguish fraudulent web pages from authentic web pages by adding certain additional characteristics.

12. FUTURE SCOPE

Phishing is a considerable problem differs from the other security threats such as intrusions and Malware which are based on the technical security holes of the network systems. The weakness point of any network system is its Users. Phishing attacks are targeting these users depending on the

trikes of social engineering. 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. APPENDIX

Github link:

https://github.com/IBM-EPBL/IBM-Project-39584-1660462497

Project demo link:

https://drive.google.com/file/d/1gGpAhsnTEp1hQfsWRy1NvBjG7nBdw4XC/view?usp=sharing

References:

- https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5

- https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-net.2020.0078