WEB PHISHING DETECTION

IBM PROJECT

Submitted by

VISHAL GUNA	**7376191CS332**
VEMBU KARTHICK T	**7376191CS326**
HEMAPRIYA R	**7376191CS176**
SABARISSHAN GK	**7376191CS267**

in a partial fulfilment for the award of the degree

BANNARI AMMAN
INSTITUTE OF TECHNOLOGY
Stay Ahead

BANNARI AMMAN INSTITUTE OF TECHNOLOGY
(An Autonomous Institution Affiliated to Anna University, Chennai)
SATHYAMANGALAM-638401

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

ABSTRACT

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. 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.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

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

1.2 PURPOSE

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.

CHAPTER 2

LITERATURE SURVEY

Survey 1:

Title: Phish-Defence: Phishing Detection Using Deep Recurrent Neural Networks

Abstract:

In the growing world of the internet, the number of ways to obtain crucial data such as passwords and login credentials, as well as sensitive personal information has expanded. Page impersonation, often known as phishing, is one method of obtaining such valuable information. Phishing is one of the most straightforward forms of cyber-attack for hackers, as well as one of the simplest for victims to fall for. It can also provide hackers with everything they need to access their targets' personal and corporate accounts. Such websites do not provide a service but instead gather personal information from users. In this paper, we achieved state-of-the-art accuracy in detecting malicious URLs using recurrent neural networks. Unlike previous studies, which looked at online content, URLs, and traffic numbers, we merely look at the text in the URL, which makes it quicker and catches zero-day assaults. The network has been optimized so that it may be utilized on tiny devices like Mobiles, Raspberry Pi without sacrificing the inference time.

Algorithm Used: Long Short-Term Memory, Gated Recurrent unit

Survey 2:

Title: Phishing website detection using machine learning and deep learning techniques

Abstract:

Phishing has become more damaging nowadays because of the rapid growth of internet users. The phishing attack is now a big threat to people's daily life and to the internet environment. In these attacks, the attacker impersonates a trusted entity intending to steal sensitive information or the digital identity of the user, e.g., account credentials, credit card numbers and other user details. A phishing website is a website which is similar in name and appearance to an official website otherwise known as a spoofed website which is created to fool an individual and steal their personal credentials. So, to identify the websites which are fraud, this paper will discuss the machine learning and deep learning algorithms and apply all these algorithms on our dataset and the best algorithm having the best precision and accuracy is selected for the phishing website detection. This work can provide more effective defenses for phishing attacks of the future.

Algorithm Used: Logistic Regression, K Nearest Neighbour, Decision Tree, Random Forest, XG Boost, Ada Boost

Survey 3:

Title: Intelligent Phishing Detection Scheme Algorithms Using Deep Learning

Abstract:

Phishing attacks have evolved in recent years due to high-tech-enabled economic growth worldwide. The rise in all types of fraud loss in 2019 has been attributed to the increase in deception scams and impersonation, as well as to sophisticated online attacks such as phishing. The global impact of phishing attacks will continue to intensify and thus a more efficient phishing detection method is required to protect online user activities. To address this need, this study focused on the design and development of a deep learning-based phishing detection solution that leveraged the universal resource locator and website content such as images, text and frames.

Algorithm Used: Convolutional Neural Network, Long Short Term Memory, Intelligent Phishing Detection System

Survey 4:

Ttitle: Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning

Abstract:

Phishing has become one of the biggest and most effective cyber threats, causing hundreds of millions of dollars in losses and millions of data breaches every year. Currently, anti-phishing techniques require experts to extract phishing sites features and use third-party services to detect phishing sites. These techniques have some limitations, one of which is that extracting phishing features requires expertise and is time-consuming. Second, the use of third-party services delays the detection of phishing sites. Hence, this paper proposes an integrated phishing website detection method based on convolutional neural networks (CNN) and random forest (RF). The method can predict the legitimacy of URLs without accessing the web content or using third-party services. The proposed technique uses character embedding techniques to convert URLs into fixed-size matrices, extract features at different levels using CNN models, classify multi-level features using multiple RF classifiers, and, finally, output prediction results using a winner-take-all approach. On our dataset, a 99.35% accuracy rate was achieved using the proposed model. An accuracy rate of 99.26% was achieved on the benchmark data, much higher than that of the existing extreme model

Algorithm Used: Random Forest, Convolutional Neural Network

2.1 EXISTING PROBLEM

The existing system uses the Classifiers, Fusion Algorithm, and Bayesian Model to detect the phishing sites. The classifiers can classify the text content and image content. Text classifier is to classify the text content and Image classifier is to classify the image content.

2.2 REFERENCES

1. Aman Rangapur, Tarun Kanakam and Dhanvanthini P, "Phish-Defence: Phishing Detection Using Deep Recurrent Neural Networks" -September 2022

2. M Selvakumari et al, "Phishing website detection using machine learning and deep learning techniques" – 2021

3. M.A.Adebowale, K.T.Lwin, M.A.Hosaain, "Intelligent Phishing Detection Scheme

Algorithms Using Deep Learning"

4. Rundong Yang, Kangfeng Zheng, Bin Wu, Chunhua Wu and Xiujuan Wang, "Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning"

2.3 PROBLEM STATEMENT DEFINITION

The problem statement is 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.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

A collaborative tool teams can use to gain a deeper insight into their customers. This Empathy map canvas discuss about what they hear about web phishing detection, What they think about web phishing detection, What they see and What they say and Do.

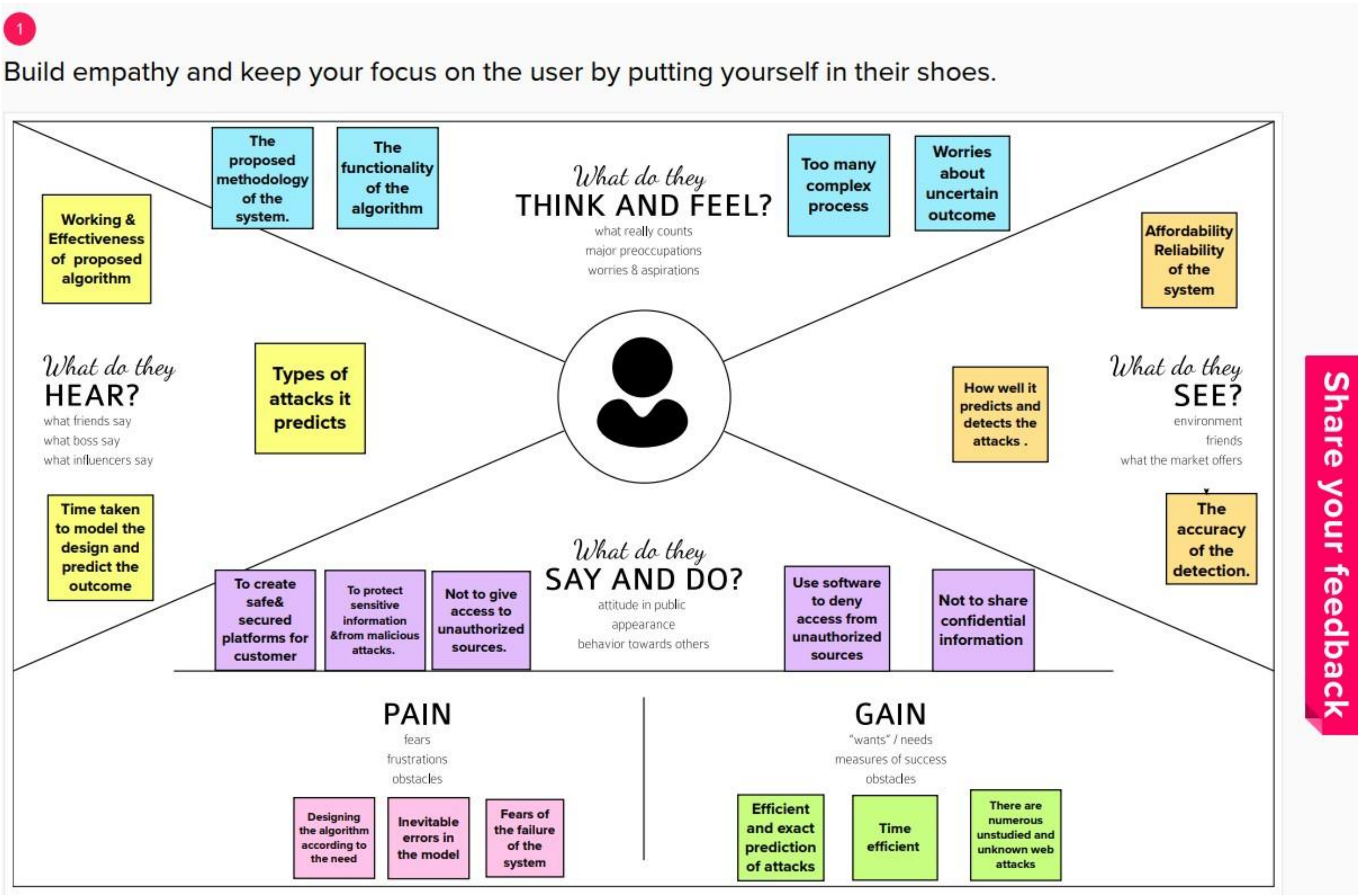


Fig 3.1 Empathy map

3.2 IDEATION & BRAINSTORMING

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

1. The victim has received a pop-up message while searching a product

2. The victim has clicked on the spoofed link that has been sent through mail

3. The victim receives a strongly worded voicemail that tends to send reply or immediately call to the phone number

4. A fraudulent message that says the recipient to change the account details immediately so that they can get the details

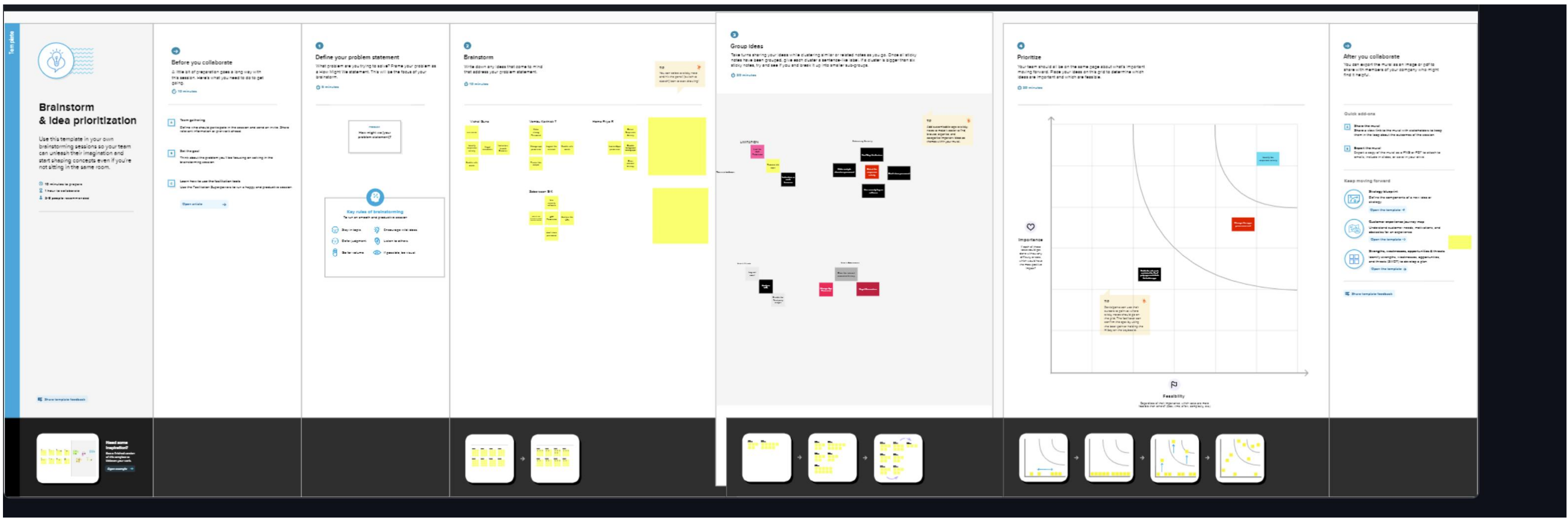


Fig 3.2 Problem statement

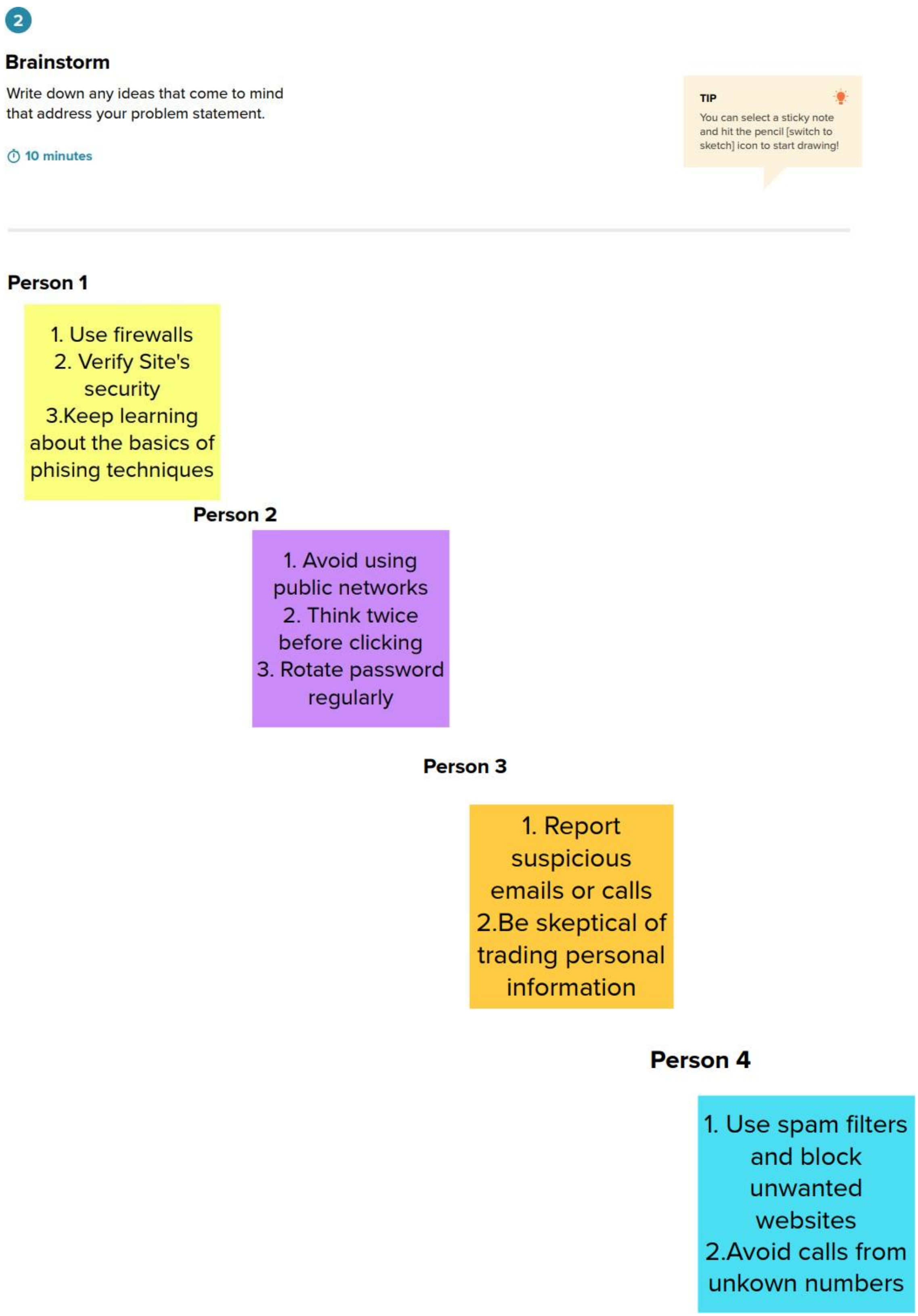


Fig 3.3 Brainstorm

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

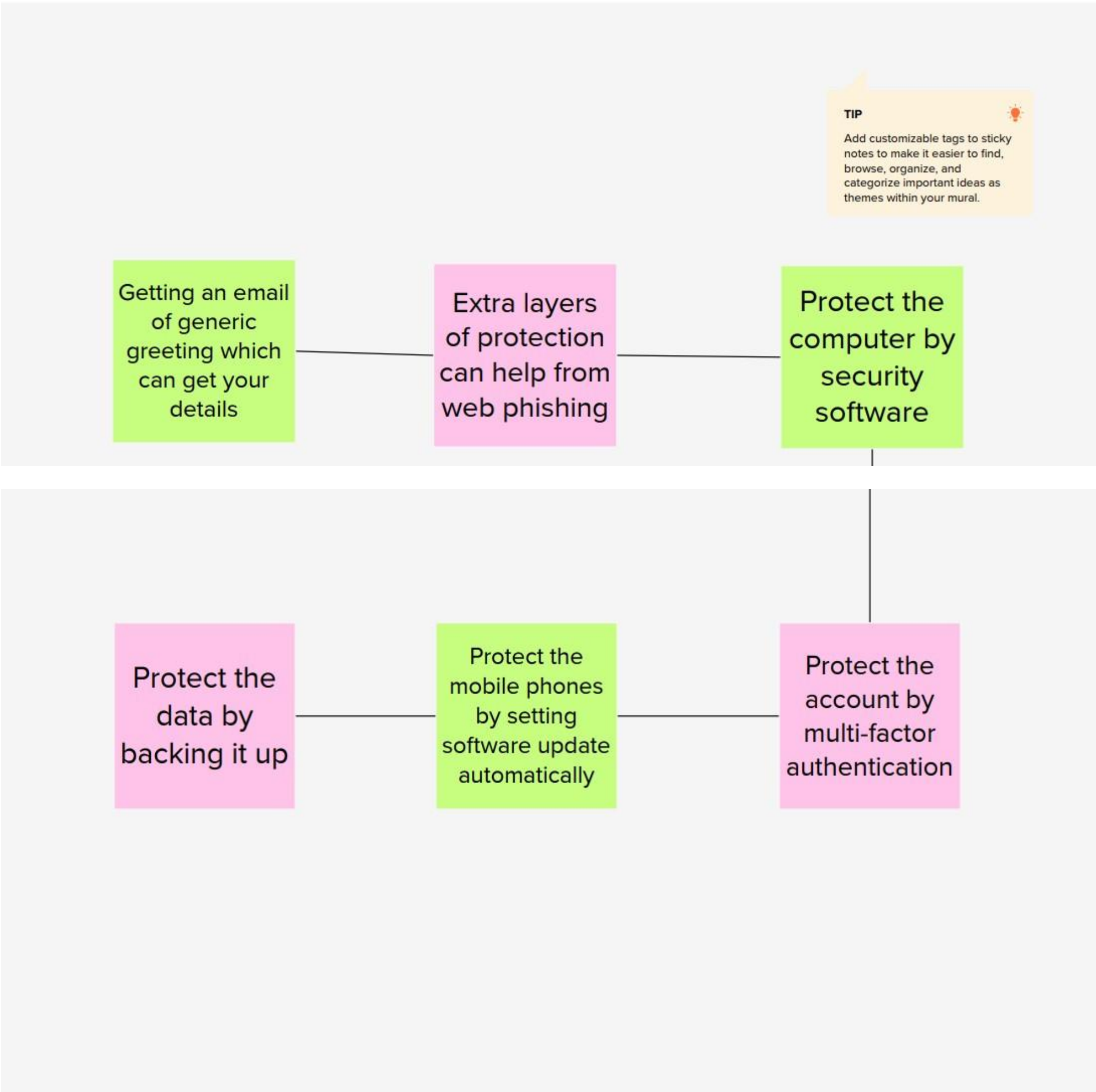


Fig 3.4 Group Ideas

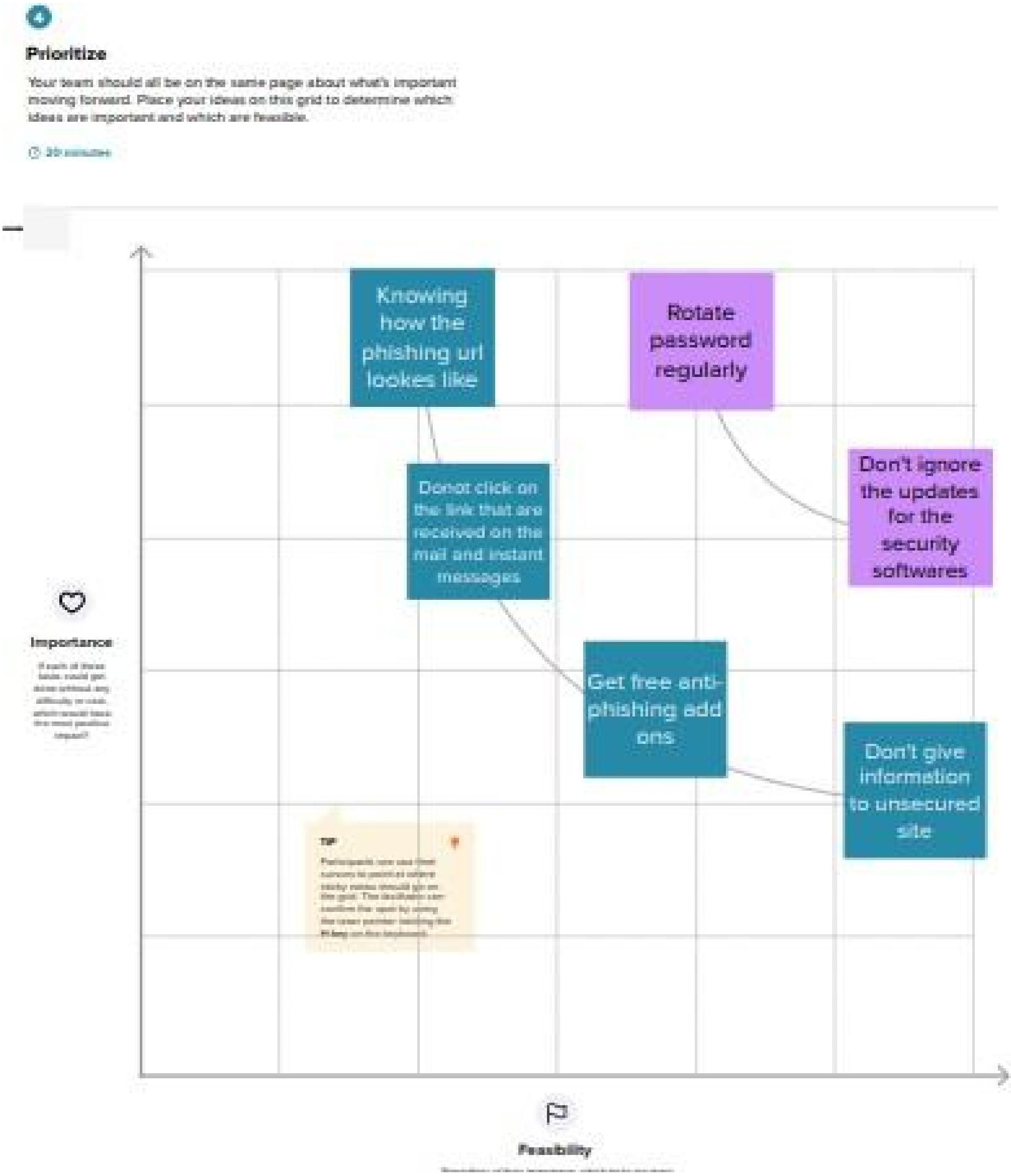


Fig 3.5 Prioritize

3.3 PROPOSED SOLUTION

The proposed solution is that, 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.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A user of the internet had to make an online purchase. He then used the internet to access the webpage. The process of displaying the goods takes time. He started to look at every item. He looks for the required products on an online page. Finally, he located the required products. He then entered all of his credit card information. password and username for making purchases via the internet. Then he got the notification "Your order" is entered successfully, and the transaction is finished. You will receive the merchandise they requested in two days. the following. He received a notification from the bank and his mobile device within 24 hours. The customer was astonished to find their account empty. Then only he realized that was a fake website and his bank

		account details was stolen by hacker .To avoid this scenario. We need to solve this problem by using the Web Phishing Detection
2.	Idea / Solution description	Every time we click on a website, an alert box stating whether it is secure or not must appear in order to combat the issue of phishing websites.The website can also be scanned to shield our computer or mobile device against phishing attacks. Although technology exists, it is still important for us as users to be aware of whether a website is secure or not. We should avoid visiting any unwelcome websites.
3.	Uniqueness	In the suggested method, the hyperlink-specific attributes were separated into 12 different categories and then used to train the machine learning algorithms. Using a dataset of phishing and non-phishing websites, we assessed how well our suggested phishing detection technique performed against several categorization algorithms.
4.	Customer Satisfaction	While utilising certain websites, an alert box will appear when we click on them, making the user aware of the website and increasing their pleasure with it. Additionally, we can scan the website to stop information from being hacked, which would increase consumer satisfaction even further.
5.	Scalability of the Solution	Thus, software-based phishing detection techniques are preferred for fighting against the phishing attack. Mostly available methods for detecting phishing attacks are blacklists/whitelists, natural language processing, visual similarity, rules, machine learning techniques

3.4 PROBLEM SOLUTION FIT

Fig 3.6 Proposed Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Functional requirements are the details and instructions that dictate how software performs and behaves. Functional requirements can vary in behaviours, features and protocols, depending on the user’s industry. Functional requirements help software engineers determine the features that support a system to operate smoothly. User can ensure that software is easy for users to operate using functional requirements. Functional requirements can help you identify the features of a system to see where you may enhance functionality. Functional requirements increase the usability of the software. A software system may include a specific feature that made the system more convenient for the users to operate.

4.2 NON – FUNCTIONAL REQUIREMENT

Non-functional requirements or NFRs are a set of specifications that describe thesystem’s operation capabilities and constraints and attempt to improve its functionality. These are basically the requirements that outline how well it will operate including things like speed, security, reliability, data integrity Nonfunctional requirements specify the quality attributes of the system, hence their second name — quality attributes.

Usability:

A system can have adequate functionality, but inadequate usability because it is too difficult to use. A usability requirement specifies how easy the system must be to use. Usability is a non-functional requirement, because in its essence it doesn't specify parts of the system functionality, only how that functionality is to be perceived by the user, for instance how easy it must be to learn and how efficient it must be for carrying out user tasks. The usability requirements must be tangible so that we are able to verify them and trace them during the development. They must also be complete so that if we fulfil them, we are sure that we get the usability we intend. Usability can predict digitswith accuracy. The model can be used in bank check processing, data entry etc

Security:

Security is a non-functional requirement assuring all data inside the system or its part will be protected against malware attacks or unauthorized access. So, the nonfunctional requirements part will set up specific types of threats that functional requirements will address in more detail. If your security relies on specific standards and encryption methods, these standards don't directly describe the behaviour of a system, but rather help engineers with implementation guides. How well are the system and its data protected against attacks. It ensures security as the uploaded image is not stored in any database.

Availability:

Availability describes how likely the system is accessible to a user at a given point in time. While it can be expressed as an expected percentage of successful requests, you may also define it as a percentage of time the system is accessible for operation during some time period. As you can see, these three metrics are closely connected. And more importantly, you should approach them together if you decide to document them as non-functional requirements for your system.

Reliability

Reliability specifies how likely the system or its element would run without a failure for a given period of time under predefined conditions. Reliability is defined as the probability that a product, system, or service will perform its intended function adequately for a specified period of time, or will operate in a defined environment without failure.

Example - The system must perform without failure in 95 percent of use cases during a month. Can process confidential information without data leakage as the data is never stored in any database. Performance improvement in fast prediction. We use Logistic regression for accurate prediction

Maintainability

Maintainability defines the time required for a solution or its component to be fixed, changed to increase performance or other qualities, or adapted to a changing environment. Like reliability, it can be expressed as a probability of repair during some time. For example, if you have 75 percent maintainability for 24 hours, this means that there's a 75 percent chance the component can be fixed in 24 hours. Maintainability is often measured with a metric like MTTRS — the mean time to restore the system.

Compatibility:

Compatibility, as an additional aspect of portability, defines how a system can coexist with another system in the same environment. For instance, software installed on an operating system must be compatible with its firewall or antivirus protection. Portability and compatibility are established in terms of operating systems, hardware devices, browsers, software systems, and their versions. For now, a cross-platform, cross-browsing, and mobile-responsive solution is a common standard for web applications.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A data flow diagram is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method. Superficially, DFDs can resemble flow charts or Unified Modelling Language, but they are not meant to represent details of software logic.

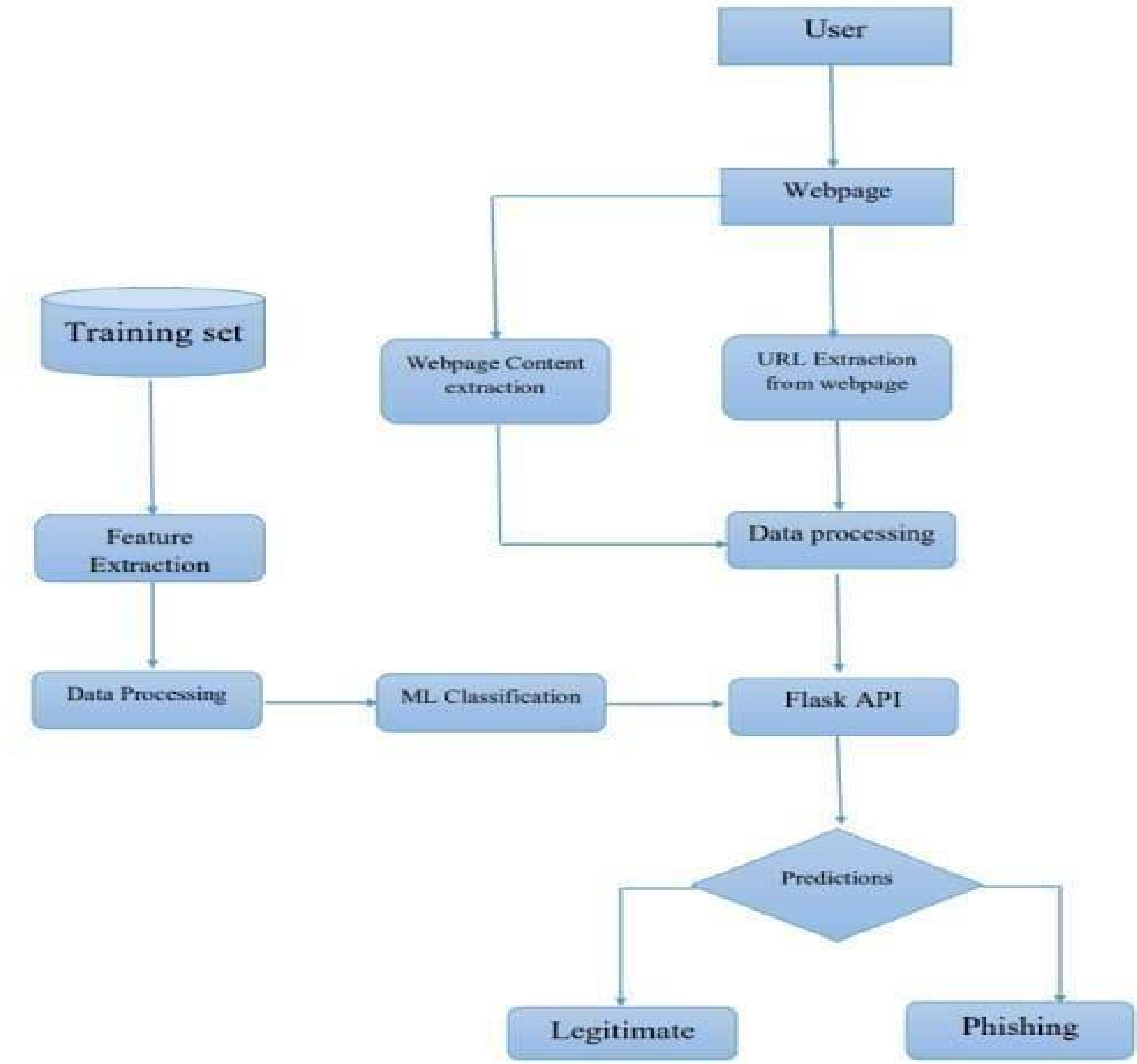


Fig 5.1 Data Flow Diagram

The data flow diagram represented in the figure 5.1 gets a URL as an input. It Scrapes feature data from the URL. Then the URL is sent to the prediction model and the website is tested whether the website is safe or not.

5.2 SOLUTION ARCHITECTURE

Solution architecture is the process of developing solutions based on predefined processes, guidelines and best practices with the objective that the developed solution fits within the enterprise architecture in terms of information architecture, system portfolios, integration requirements and many more.

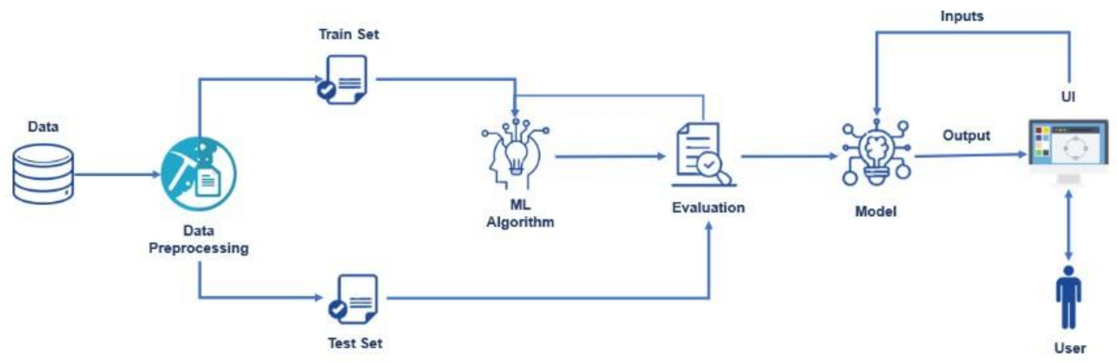


Fig 5. 2 Solution Architecture

The architecture explained above is the data is pre processed and it is divided into training set and testing set. Then the model is processed through a machine learning algorithm and it is send for the process of evaluation. The model is built using the flask and html and an user interface is produced for the user to use. The user gives the input to the model and the output is produced to the user.

5.3 USER STORIES

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

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 confirmingmy 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.	I this i will find the correct classifier for producing the result	Medium	Sprint-2

Table 5.1 User Stories

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT & ESTIMATION PLANNING

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Vishal Guna
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Hema priya R
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	Vembu Karthick K
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Sabarisshan Gk
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	Vishal Guna

Table 6.1 Sprint estimation Table

6.2 SPRINT DELVERY SCHEDULE

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

Table 6.2 Sprint delivery schedule

CHAPTER 7

CODING AND SOLUTIONING

7.1 LOGISTIC REGRESSION

This type of statistical model (also known as *logit model*) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

Logit(pi) = 1/(1+ exp(-pi))

ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

In this logistic regression equation, logit(pi) is the dependent or response variable and x is the independent variable. The beta parameter, or coefficient, in this model is commonly estimated via maximum likelihood estimation (MLE). This method tests different values of beta through multiple iterations to optimize for the best fit of log odds. All of these iterations produce the log likelihood function, and logistic regression seeks to maximize this function to find the best parameter estimate. Once the optimal coefficient (or coefficients if there is more than one independent variable) is found, the conditional probabilities for each observation can be calculated, logged, and summed together to yield a predicted probability. For binary classification, a probability less than .5 will predict 0 while a probability greater than 0 will predict 1. After the model has been computed, it's best practice to evaluate the how well the model predicts the dependent variable, which is called goodness of fit. The Hosmer–Lemeshow test is a popular method to assess model fit.

7.2 THE USER INTERFACE

The UI is built using HTML, CSS and JavaScript. The backend of the application is handled by Flask, a python framework for backend application development. The application consists of two pages,

1. Index.html
2. Style.css
3. Mainpage.Html

Info of Website

A URL based phishing attack is carried out by sending malicious links, that seems legitimate to the users, and tricking them into clicking on it. In phishing detection, an incoming URL is identified as phishing or not by analysing the different features of the URL and is classified accordingly. Different machine learning algorithms are trained on various datasets of URL features to classify a given URL as phishing or legitimate.

Machine Learning Algorithms

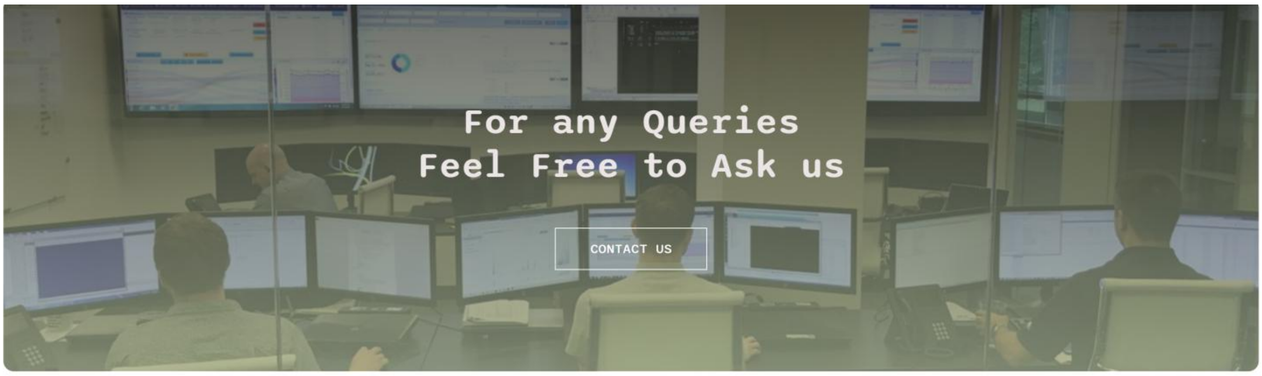
There are several machine learning algorithms such as Naive Bayes, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression and K-Nearest-Neighbor for detecting phishing websites. This is a very popular approach that ha proved to be very efficient and accurate compared to other methods.

Used Algorithms

Naive Bayes, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression and K-Nearest-Neighbor for detecting phishing websites. This is a very popular approach that ha proved to be very efficient and accurate compared to other methods.

Approaches

In List Based approach, there are two lists, called whitelist and blacklist to classify legitimate and phishing URLs respectively. In [5], access to websites takes place only if the URL is in the whitelist. In [6] blacklist is used. In Heuristic Based approach, the structure of a phishing URL is analysed. A pattern of URLs that were previously classified as phishing is created. URLs are classified according to their compliance with this pattern. The methods used to process the features of the URL plays a significant role in classifying websites accurately .



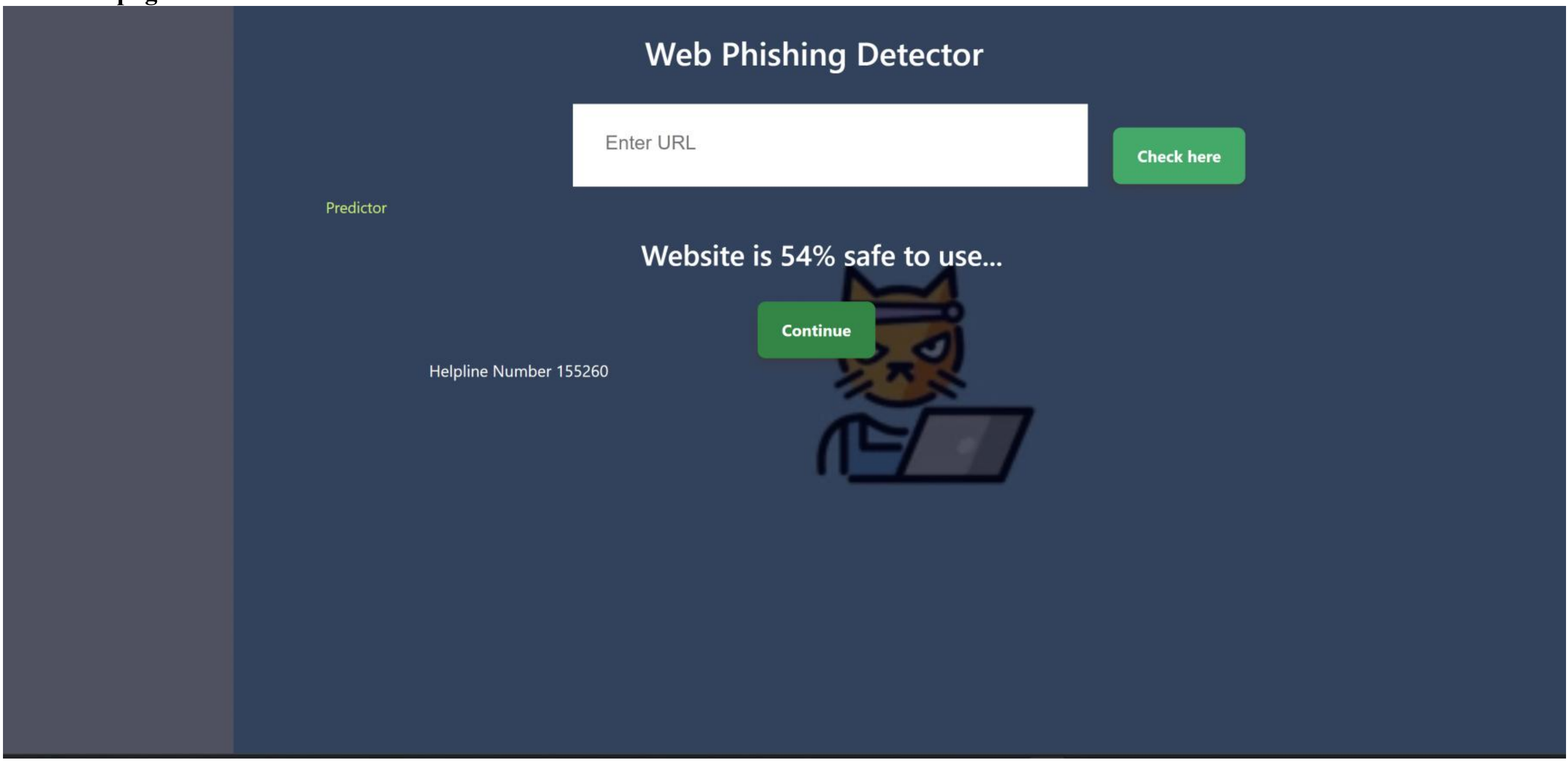
About Us

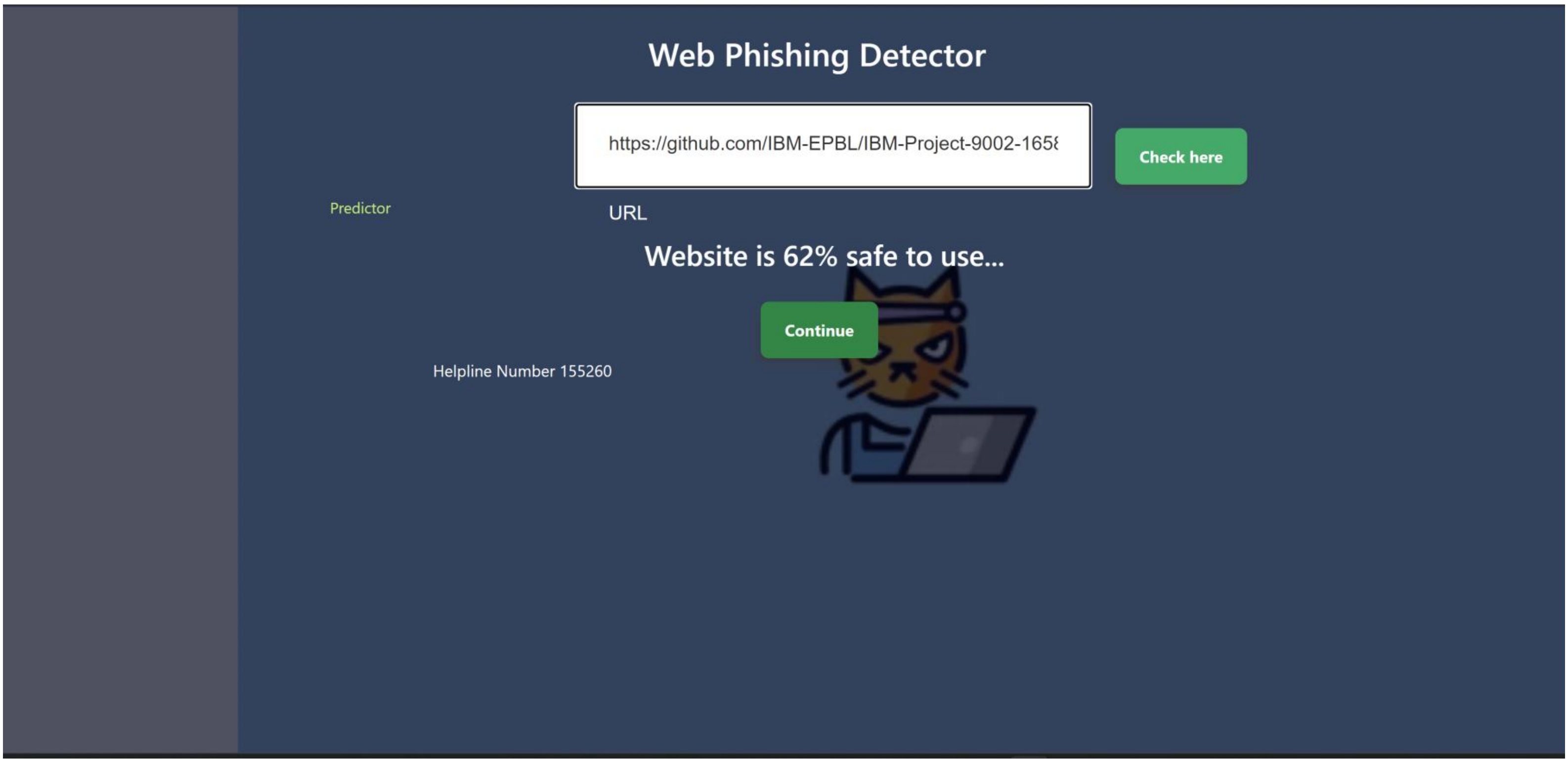
Phishing works by sending messages that look like they are from a legitimate company or website. The message will usually contain a link that takes the user to a fake website that looks like the real thing. The user is then asked to enter personal information, such as their credit card number. We are now currently working on the identifying process .In future , you hope that we come up with the best solution.

Made With Pleasure❤By Team IBM-PNT2022MID01661

OUTPUT

Predictor page :





CHAPTER 8

RESULT

8.1 PERFORMANCE METRICS

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.

The performance metrics of the model -01 is 62% safe to use site

The performance metrics of the model -02 is 58% unsafe to use the site

CHAPTER 9

ADVANTAGES AND DISADVANTAGES

9.1 ADVANTAGES AND DISADVANTAGES

ADVANTAGES

A mailbox-level anti-phishing solution offers an additional layer of protection by analyzing account information and understanding users' communication habits. This delivers an enhanced level of phishing protection to detect attacks faster, alert users and remediate threats as quickly as possible. Machine learning scores sender reputation enabling a baseline for what "normal communications" with a user should look like. It can then compare correspondence and incoming messages with multiple data points to identify and learn from anomalies.

DISADVANTAGES

Although our proposed approach has attained outstanding accuracy, it has some limitations. First limitation is that the textual features of our phishing detection approach depend on the English language. This may cause an error in generating efficient classification results when the suspicious webpage includes language other than English. About half (60.5%) of the websites use English as a text language. However, our approach employs URL, noisy part of HTML, and hyperlink based features, which are language-independent features. The second limitation is that despite the proposed approach uses URL based features, our approach may fail to identify the phishing websites in case when the phishers use the embedded objects (i.e., Javascript, images, Flash, etc.) to obscure the textual content and HTML coding from the anti-phishing solutions. Many attackers use single server-side scripting to hide the HTML source code. Based on our experiments, we noticed that legitimate pages usually contain rich textual content features, and high amount of hyperlinks (At least one hyperlink in the HTML source code). At present, some phishing webpages include malware, for example, a Trojan horse that installs on user's system when the user opens the website.

Hence, the next limitation of this approach is that it is not sufficiently capable of detecting attached malware because our approach does not read and process content from the web page's external files, whether they are cross-domain or not. Finally, our approach's training time is relatively long due to the high dimensional vector generated by textual content features. However, the trained approach is much better than the existing baseline methods in terms of accuracy.

CHAPTER 10

CONCLUSION

10.1 CONCLUSION

A systematic review of current trends in web phishing detection is carried out and a taxonomy of web phishing detection is proposed based on the input parameters chosen. The performance of the state-of-the-art web phishing detection approaches is evaluated and presented in detail. This paper also discussed the limitations of the existing web phishing detection techniques for future research direction. The analysis given in this paper will help the academia and industries to acknowledge the current status of web phishing detection and lead them to come up with new ideas to develop the best web anti-phishing technique.

CHAPTER 11

FUTURE SCOPE

11.1 FUTURE WORK

In future work, we plane to include some new features to detect the phishing websites that contain malware. As we said in “Limitations” section, our approach could not detect the attached malware with phishing webpage. Nowadays, blockchain technology is more popular and seems to be a perfect target for phishing attacks like phishing scams on the blockchain. Blockchain is an open and distributed ledger that can effectively register transactions between receiving and sending parties, demonstrably and constantly, making it common among investors. Thus, detecting phishing scams in the blockchain environment is a defiance for more research and evolution. Moreover, detecting phishing attacks in mobile devices is another important topic in this area due to the popularity of smart phones, which has made them a common target of phishing offenses.

CHAPTER 12

APPENDICES

SOURCE CODE

app.py

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

import _json


import requests
import feature
from feature import FeatureExtraction


# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "<j8gnwjgK1Pk7vrnA7wND_ivnq17Bh4dlbFy3-iq_s-mt>"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
url = request.form['url']
        obj = feature.FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        print(x.tolist())

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [["index", "having_IPhaving_IP_Address", "URLURL_Length",
"Shortining_Service","having_At_Symbol",  "double_slash_redirecting",   "Prefix_Suffix",
"having_Sub_Domain","SSLfinal_State","SSLfinal_State", "Domain_registeration_length","Favicon","port",   "HTTPS_token",
"Request_URL", "URL_of_Anchor",  "Links_in_tags",  "SFH", "Submitting_to_email", "Abnormal_URL",   "Redirect",
"on_mouseover",   "RightClick",  "popUpWidnow", "Iframe",  "age_of_domain",   "DNSRecord",   "web_traffic", "Page_Rank",
"Google_Index",   "Links_pointing_to_page",  "Statistical_report"
        ]], "values": x.tolist()}]}
response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/wpd/predictions?version=2022-
11-22', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())

response_text = response_scoring.json()
y_pred = response_text['predictions'][0]['values'][0][0]
```

```
y_pro_phishing = response_text['predictions'][0]['values'][0][1][0]
y_pro_non_phishing = response_text['predictions'][0]['values'][0][1][1]

# load model
class Flask:
    pass


app = Flask(__name__)
model = pickle.load(open('Phishing_website.pkl', 'rb'))


# fetches the URL given by the URL and passes to feature
@app.route('/_predict', methods=['POST'])
def y_predict():
    '''
        For rendering results on HTML GUI
        '''


    url = request.form['URL']
    checkprediction = feature.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 = "Your are on the wrong site. Be cautious!"
    return render_template('final.html', prediction_text='{}'.formate(pred), url=url)


# Takes the input parameters fetched from the URL by inputsScript and returns the
@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)
```

feature.py

```
from base64 import urlsafe_b64decode, urlsafe_b64encode
import ipaddress
import re
from urllib import response
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
from sympy import Domain
import urllib3
import whois
from datetime import date, datetime
from urllib.parse import urlparse


class FeatureExtraction:
    features = []

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

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

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

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

        self.features.append(self.having_IPhaving_IP_Address())
        self.features.append(self.URLURL_Length())
        self.features.append(self.Shortining_Service())
        self.features.append(self.having_At_Symbol())
        self.features.append(self.double_slash_redirecting())
        self.features.append(self.Prefix_Suffix())
        self.features.append(self.having_Sub_Domain())
        self.features.append(self.SSLfinal_State())
        self.features.append(self.Domain_registeration_length())
        self.features.append(self.Favicon())

        self.features.append(self.port())
        self.features.append(self.HTTPS_token())
        self.features.append(self.Request_URL())
        self.features.append(self.URL_of_Anchor())
        self.features.append(self.Links_in_tags())
        self.features.append(self.SFH())
        self.features.append(self.Submitting_to_email())
        self.features.append(self.Abnormal_URL())
        self.features.append(self.Redirect())
        self.features.append(self.on_mouseover())

        self.features.append(self.RightClick())
        self.features.append(self.popUpWidnow())
        self.features.append(self.Iframe())
        self.features.append(self.age_of_domain())
        self.features.append(self.DNSRecord())
        self.features.append(self.web_traffic())
        self.features.append(self.Page_Rank())
        self.features.append(self.Google_Index())
        self.features.append(self.Links_pointing_to_page())
        self.features.append(self.Statistical_report())

    # 1.having_IPhaving_IP_Address

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

```python
        return 1

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

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

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

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

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

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

```python
        return -1

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

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

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

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

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

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

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

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

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

```
                i = i+1

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

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

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

    # 14. URL_of_Anchor
    def URL_of_Anchor(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 (urllib3 in a['href'] or
self.domain in a['href']):
                    unsafe = unsafe + 1
                i = i + 1

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

```python
        except:
            return -1

    except:
        return -1

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

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

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

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

# 16. SFH
def SFH(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
```

```python
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
                else:
                    return 1
    except:
        return -1

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

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

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

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

```
            return -1
    except:
        return -1

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

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

# 23. Iframe
def Iframe(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
        return -1

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

        today = date.today()
```

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

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

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

# 26. web_traffic
def web_traffic(self):
    try:
        rank = BeautifulSoup(urllib.request.urlopen(
            "http://data.alexa.com/data?cli=10&dat=s&url=" + urlsafe_b64decode).read(), "xml").find("REACH")['RANK']
        if (int(rank) < 100000):
            return 1
        return 0
    except:
        return -1

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

        global_rank = int(re.findall(
```

```
            r"Global Rank: ([0-9]+)", prank_checker_response.text)[0])
        if global_rank > 0 and global_rank < 100000:
            return 1
        return -1
    except:
        return -1

# 28. Google_Index

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

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

# 30. Statistical_report
def Statistical_report(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',
urlsafe_b64encode)
        ip_address = socket.gethostbyname(self.domain)
        ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.
174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

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

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

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.7
7|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

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

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.1
57\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1

    def getFeaturesList(self):
        return self.features
```

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Phishing Detector</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<meta charset="UTF-8">
<style>
    @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
```

```
}



body{
    scroll-behavior: smooth;
}
header{
    width:100%;
    display:flex;
    justify-content: space-between;
    padding:10px 10px;
    flex-direction: column;
    align-items: center;
    position:fixed;
    z-index:999;
}
html{
    scroll-behavior: smooth;
}


body{
    font-size: 100%;
    font-family: sans-serif;
}


div, section, span, ul, li, a, header,h1{
    box-sizing: border-box;
}
.joint{
    width:100%;
    display:flex;
    flex-direction:row;
}
.result{
    width:500px;
    border:2px solid black;
    margin-left:20px;
    text-align:center;
    justify-content: center;
    align-items: center;
    margin-top:10%;
    padding:5%;
}


h4{
```

```css
    text-decoration: underline;
    color:rgb(76, 76, 202);
}

.sec-container{
    width:450px;
    height:400px;
    background-color: #678efa;
    border-radius: 2%;
    display:flex;
    align-items: center;
    flex-direction: column;
    border: 1px solid;
    box-shadow: 5px 10px #adacac;

}
.sec-1-container{
    width:550px;
    height:250px;
    background-color: #678efa;
    border-radius: 2%;
    display:flex;
    align-items: center;
    flex-direction: column;
    border: 1px solid;
    margin-top:10%;
    margin-left:10%;
    padding:2%;
    box-shadow: 5px 10px #adacac;
}
.sec-container h2,p{
    align-items: center;
    position:relative;
    text-align: center;
    padding:5%;
}
.frst-container h2,p{
    align-items: center;
    position:relative;
    text-align: center;
    padding:5%;
}
.frst-container{
    width:450px;
    height:400px;
```

```
    margin-right:10%;
    border: 1px solid;
  box-shadow: 5px 10px #888888;
    background-color: #fff;
    border-radius: 2%;
    display:flex;
    align-items: center;
    flex-direction: column;
    padding-right:4%;
}

#menu li{
    list-style: none;
    display: inline-block;
    margin: 20px;
}

#menu li a{
    color: black;
    text-decoration: none;
    font-size: 20px;
}

section{
    width: 100%;
    float: left;
    height: 100vh;
    position: relative;
}
.head{
    color:black;
    display:flex;
    margin-left:5%;
    text-decoration: none;

    font-size:20px
}
#home{
    background-color: #678efa;
}
.img-share{
    width:75%;
    height:500px;
    position:relative;
    float:right;
```

```
    margin-top:5%;
    /* margin-left:8%; */
    /* border: 2px solid black; */
    /* margin-right:60%; */
    background-image: url('../static/bg2.png');
}
.frst-container img{
    width:100px;
    height:100px;
}
.wordings{
    width:280px;
    display: flex;
    margin-top: 20%;
    margin-left:2%;
    flex-direction: column;
}
.highlight{
    font-size: 30px;
    color:#ffb703
}
.wordings p{
    color:#fff;
}
.join{
    display: flex;
    flex-direction:row;
}
#about{
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    background-color: rgb(110, 136, 89);
    /* background-color: #4070f4; */
}

#product{
     background-color: #B52B65;
}

#help{
     background-color: #678efa;
}
```

```css
#contact{
     background-color: rgb(63, 109, 236);
}


h1{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    color: #fff;
}
.container{
    position: relative;
    max-width: 430px;
    width: 100%;
    background: #fff;
    border-radius: 10px;
    box-shadow: 0 5px 10px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    margin: 0 20px;
}
button{
    margin-top:60px;
    display:flex;
    flex-direction:column;
    align-items:center;
    /* margin-left:100px; */
    width:150px;
    height:50px;
    justify-content: center;
    border-radius: 6%;
    background-color:#fff ;
    border:none;
    color: black;
    font-size: 16px;
    font-weight: 500;


}
.second-container{
    display: flex;
    width:100%;
    height:100%;



}
```

```css
.navbar{
    border:2px solid rgb(89, 96, 88);
    width:1200px;
    background-color: rgb(241, 246, 235);
    border-radius: 30px;
    margin-top:2%;
    overflow: hidden;
    margin-left:2%;
    z-index: 999;

}

.navbar ul{
    overflow: auto;
}
.navbar li{
    float:center;
    list-style: none;
    margin: 10px 10px;

}
.navbar li a{
    padding: 5px 30px;
    text-decoration: none;
    color: black;
}
.navbar li a:hover{
    color: red
}
.search{
    float: right;
    color: white;
    padding: 12px 75px;
}
.navbar input{
    border: 2px solid black;
    border-radius: 14px;
    padding: 3px 17px;
    width: 150px;
}
.sec-1-container input{
    width:400px;
    height:60px;
    padding:5%;
}
```

```
#contact div{
    width:500px;
    height: 480px;
    margin-top:5%;
    margin-left: 32%;
    padding:5%;
}
input[type=text], select, textarea {
    width: 100%; /* Full width */
    height:27px;
    /* padding: 5px; Some padding  */
    border: 1px solid #ccc; /* Gray border */
    border-radius: 4px; /* Rounded borders */
    box-sizing: border-box; /* Make sure that padding and width stays in place */
    /* margin-top: 2px; Add a top margin */
    /* Bottom margin */
    resize: vertical /* Allow the user to vertically resize the textarea (not horizontally) */
  }
  .point{
    text-decoration: none;
    color:#fff;
    margin-left:30%;
  }
  /* Style the submit button with a specific background color etc */
  input[type=submit] {
    background-color: #ccc;
    color: white;
    padding: 8px 10px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }

  /* When moving the mouse over the submit button, add a darker green color */
  input[type=submit]:hover {
    background-color: #d7edc7;
  }

  /* Add a background color and some padding around the form */
  .container {
    border-radius: 5px;
    background-color: #f2f2f2;
    padding: 8px;
  }
```

```html
   </style>
</head>
<body>
  <nav class="navbar">
          <ul id="menu">
              <li><a  href="C:\Users\mathe\OneDrive\Desktop\Web Phishing Detection\MainPage.html">Home</a></li>
              <li><a  href="C:\Users\mathe\OneDrive\Desktop\Web Phishing Detection\MainPage.html">About</a></li>

          </ul>

   </nav>
   <div class="joint">

<div class="sec-1-container">

   <form action = "/predictdata" method="post">
       <label>Check Your Link Here</label>
       <input type="text" name="url">
      <button class="button21" role="button" type="submit">Confirm</button>
   </form>
  </div>

<div class="result">
<h2 style="color:black">
   {{predicted}}</h2>
   <h4>{{url}}</h4>
</div>
</div>
 <!-- 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.js"
 integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
 crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
 integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
 crossorigin="anonymous"></script>



<script>

    let x = '{{xx}}';
```

```
    //console.log(rt);
    // debugger
    let num = x*100;
    if (0<=x && x<0.50){
        num = 100-num;
    }
    let txtx = num.toString();
    if (!(x=="" || x==" " || x==undefined || x==NaN)){
            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";
                flag=true;
            }else if (0<=x && x<0.50){
                var label = "Website is "+txtx +"% unsafe to use..."
                document.getElementById("prediction").innerHTML = label ;
                //document.getElementById("button1").style.display="none";
                document.getElementById("button2").style.display="block";
                flag=true
            }
            document.getElementsByClassName("button")[0].style.display="none";
            document.getElementsByClassName("button21")[0].style.display="block";


        }

</script>


<p>Helpline Number <i class="fa fa-phone" ></i> 155260</p>






</body>
</html>
```

GitHub Link:https://github.com/IBM-EPBL/IBM-Project-9002-1658942080

Demo-Video Link:https://drive.google.com/drive/folders/1ys0a7RbJSPlzjEMwQijFVDuyarSMFOjR?usp

=share_link