Web Phishing Detection

A.KIRTHIC VISHNU	19Z301
ADHARSH.S	19Z302
KUMARESH.S	19Z327
MRIDULA.M	19Z332

Team ID : PNT2022TMID12640

NAALAIYA THEERAN PROJECT

Industry Mentor: Sandesh P

Faculty Mentor: Vani K

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

NOVEMBER 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Numerous consumers use e-banking to pay for their online purchases of goods. There are several e-banking websites that frequently request sensitive information from users—such as usernames, passwords, and credit card information—for harmful purposes. Phishing websites are this kind of online banking website. One of the most important software services for communications on the Internet is the web service. One of the many security risks to web services on the Internet is web phishing.

We suggested an intelligent, adaptable, and successful system that is built on applying classification algorithms to identify and anticipate e-banking phishing websites. To classify the authenticity of the phishing datasets, we used classification algorithms and approaches to extract the relevant characteristics. The final phishing detection rate can be used to identify the e-banking phishing website based on some key features like URL and domain identity, security and encryption standards, and other factors. Our system will use a data mining algorithm to determine whether an e-banking website is a phishing website or not after a user completes an online transaction and makes a payment through it.

1.2 PURPOSE

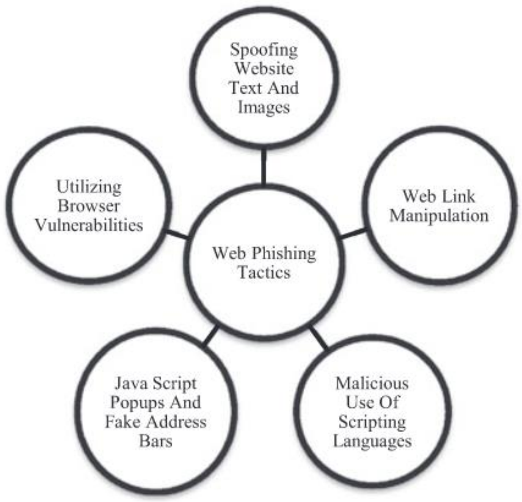
A phishing site recognition system is the working of a machine to train itself to recognize a phishing URL from different sources such as emails, images, articles, etc for online banking portals to recognize malicious URLs to protect users from getting their sensitive credentials being stolen

Web phishing detection system is the process to **utilize the ability of machines to recognize phishing URLs**. It is not an easy task for the machine because there are a lot of different factors that need to be considered to classify any URL as malicious.

2. LITERATURE SURVEY

1. A survey and classification of web phishing detection schemes [1]

- The paper highlights the type of attacks possible taking various vectors into consideration and classifies them as shown in the figure below


Spoofing Website Text And Images
Utilizing Browser Vulnerabilities
Web Link Manipulation
Web Phishing Tactics
Java Script Popups And Fake Address Bars
Malicious Use Of Scripting Languages

- It also highlights various phishing preventive solutions based on watermarking, RFID technology, QR code based,etc. The paper also mentions various detection schemes and highlights their pros and cons like:
    - ***Search engine-based techniques:***
      These techniques collect text, photos, or URLs from websites and utilize them as search terms to gauge how popular a website is in order to identify phishing. The methods differ, though, in terms of I the type and quantity of features extracted from a webpage (text, URL, or images), (ii) the number of search engines used to gauge the popularity of the webpage, (iii) the number of top results used for matching,
    - ***Heuristics and machine learning-based techniques:***
      This group of techniques extracts a number of features, such as web page content, URLs, and/or network information, and a set of machine learning or classification algorithms, which are then used to build a model for classification.
- The study offers an analysis of the methods suggested for phishing detection that has been carried out. The study focuses on the fact that phishing detection methods outperform phishing prevention and user education strategies since they don't call for modifications to authentication systems and don't rely on the user's capacity to recognize phishing. Regarding the additional hardware needed and password management, phishing detection solutions are additionally less expensive than phishing prevention solutions.
- Because they just need a single search engine query result and its underlying algorithm to identify phishing websites at the user's end, search engine-based strategies have been shown to be the lightest solutions for phishing detection.

2. Intelligent web-phishing detection and protection scheme using integrated features of Images, frames, and text [2]

The combined features of the text, graphics, and frames are used in this paper's robust adaptive neuro-fuzzy inference system (ANFIS)-based web-phishing detection and protection approach. The integrated elements of phishing websites' graphics, frames, and text were used in this work to propose an intelligent phishing detection and defense strategy. Based on the plans outlined in Aburrous et al. (2010) and Barraclough et al. (2014), an effective ANFIS algorithm was created, examined, and verified for phishing website identification and protection (2015). The study employs a hybrid technique to choose text features, utilizing factors such as Page rank, Google Index, long URLs, domain entity relationships, and many others.

3. Intelligent cyber-phishing detection for online [3]

- The approach is based on multiple ML classification algorithms, using ANFIS implemented in MATLAB toolbox with features and NB, PART, J48, JRip implemented in Waikato Environment Knowledge Analysis (WEKA).
- Methodology combines blacklist-based features, web content-based and heuristic-based approaches enabling a set of data (phishing websites, suspicious websites, spoofed web and legitimate websites) to be extracted from diverse sources.
- Based on evaluation of proposed methodology using ANFIS and MATLAB, using randomised and time-based evaluation, the method achieved 98.1% accuracy with 1.9% average testing error. Upon fine tuning, 99% accuracy was achieved with a 0.1% average error. Time taken to build the model was 0.01 Secs. The results demonstrate that the method can generalise well to new phishing attack
- J48 classifier shows a good performance that is 99.3% instances are correctly classified (TP), while 0.66% instances are incorrectly classified. Time taken to build the model is 0.01 secs
- NB classifier, on applying Randomised and Time-based evaluation methods, has high performance. 99.3% phishing instances correctly classified (TP) and 0.66% instances are incorrectly classified, while Time taken to build the model was 0.01 Secs. Recall maintained the accuracy, while F-Measure average score slightly decreased by 0.03%. ROC curve average score decreased further by 0.5%, Precision score was the lowest with 98.7% accuracy. The overall accuracy (99.3%) exceeds 95%
- PART classifier randomly split the features equally into Tens, trained on training-sets and test on unseen test-set. t PART algorithm achieved high performance, obtaining 99.3% accuracy (correctly classified as TP) and 0.66% instances incorrectly classified. Time taken to build the model was 0.01 to 0.006 Secs, which was the best speed
- JRip demonstrated best performance, obtaining 99.3% accuracy with speed of 0.07 to 0.14 Secs with 0.66% instances incorrectly classified. This is the worst achievement in speed
- The proposed novel methodology identifies abnormal features in URLs, links and images, text, forms, frames links and files. The results show that the approach can also

classify between phishing, suspicious and legitimate websites accurately
- The method with 0.6 error rates could be improved by deploying constant flow of emerging features specifically dedicated to detecting fraudulent websites. Whilst this method has produced excellent results, it is important to have continuous features update in order to keep ahead of evolving phishing strategies

4. Boosting the Accuracy of Phishing Detection with Less Features Using XGBOOST [4]

- Use of Anti-Phishing and Network analysis tool (CANTINA) heuristic features with a new additional attribute that are combined with the new ones (Domain Top page Similarity) to detect Phishing pages
- Definition of a set of rules in order to identify Phishing web pages, Two groups of rules were distinguished: -the simple rules, based on web page URL, and the higher complex and time-consuming rules based on the analysis of metadata, query search engines, and blacklists the search engine based rules, the red aged keywords based rules, the obfuscation based rules, the blacklists based rules, the reputation-based rules, the content-based rules.
- Extraction of identities from some features such as Meta title, meta description, content attributes, and "href" attributes of tag for detecting Phishing attacks.
- Use of feature vector of size 23, in which four were structural features from URLs, nine were lexical features and ten were features targeting mostly brands and websites and classification using SVM.
- Detection of hidden URLs using lexical features
- Use of lexical and domain features extracted from Phishing URLs to detect Phishing websites
- Use of Tabsol to fight against Tab nabbing (Tab nabbing is a variant of Phishing attack in which a malicious page opened in a tab, disguises itself to a popular website's login page such as Gmail, or Facebook login pages with the objective to steal credentials)
- Comparison of different features assessment techniques in the website phishing context to determine the minimal set of features for detecting phishing activities
- Experiment shows that the comparison of the accuracy of algorithms for Different Feature Groups based on the decisive values of the features demonstrated that the best accuracy is obtained for Random Forest by 96.07%, but random forests have been observed to overfit some datasets and noisy classification tasks.
- Use of Probabilistic Neural Networks (PNNs) and e-integration of PNN with K-medoids clustering to significantly reduce complexity without jeopardizing the detection accuracy. The experimental results show that 96.79% accuracy is achieved with low false errors.
- Use of the XGBOOST algorithm which improved the performance that a predictive model can achieve in the task of phishing website detection
- Comparison of XGBOOST with Probabilistic Neural Networks (PNN) and Random forest (RF) method in which all the methods (classifiers) were trained and tested using the same dataset and evaluated using the same performance metrics for a fair comparison, XGBOOST algorithm returned the accuracy of 97.27%

5. A Deep Learning Technique for Web Phishing Detection Combined URL Features and Visual Similarity [5]

- Listed ways in which phishing detection is usually performed
    1. DOM (Document Object Model) tree analysis
    2. Visual feature-based technique
    3. CSS (Cascading Style Sheets) based similarity analysis
    4. Website image comparison
    5. Visual perception method
    6. Hybrid Method
- Use of Convolutional Neural Networks (CNNs) to detect web phishing attacks based on URLs and screenshots of websites, as CNN is best in processing high dimensional data such as videos and images
- Division of snapshots of legitimate and suspected pages into blocks and matching using Earth Mover's algorithm, which gives a high detection rate (99.6%)
- Normalized Compression Distance (NCD) to compute similarities based on the distance between the image of a requested website and the image of a cached benign website, which gives a high true positive rate but is practically impossible with real-time browsing
- Formulation of web phishing detection as a binary classification problem with the URL and images of websites as input leading to their classification as either legitimate websites or phished websites which can detect newly created phishing webpages based only on the URL and the screenshot of suspicious websites. The proposed model shows a classification accuracy of 99.67%

2.1 EXISTING PROBLEM

Many customers use e-banking to pay for their online purchases of goods. Some e-banking websites, for malicious content, request users to submit private information like usernames, passwords, credit card numbers, and more. Phishing websites are this kind of online banking website. One of the most vital software services for communications over the Internet is the web service. One of the many security risks to web services on the Internet is web phishing.

2.2 REFERENCES

[1] Varshney, G., Misra, M., and Atrey, P. K. (2016) A survey and classification of web phishing detection schemes. *Security Comm. Networks*, 9: 6266– 6284. doi: 10.1002/sec.1674.

[2] M.A. Adebowale, K.T. Lwin, E. Sánchez, M.A. Hossain, Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text, Expert Systems with Applications, Volume 115, 2019, Pages 300-313, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2018.07.067.

[3] Barraclough, P. A., Fehringer, G., & Woodward, J. (2021). Intelligent cyber-phishing detection for online. Computers & Security, 104, 102123. https://doi:10.1016/j.cose.2020.102123

[4] Hajara, Musa & Gital, A & Mohzo, & Bitrus, Gideon & Jumaat, Nurul & Juma'at, & Muhammad, & Balde, Abubakar. (2020). Boosting the accuracy of phishing detections with less features using XGBOOST. 8. 81-90. https://www.researchgate.net/publication/339676208_Boosting_the_accuracy_of_phishing_detections_with_less_features_using_XGBOOST

[5] Al-Ahmadi, Saad, A Deep Learning Technique for Web Phishing Detection Combined URL Features and Visual Similarity (2020). International Journal of Computer Networks & Communications (IJCNC) Vol.12, No.5, September 2020, Available at SSRN: https://ssrn.com/abstract=3716033

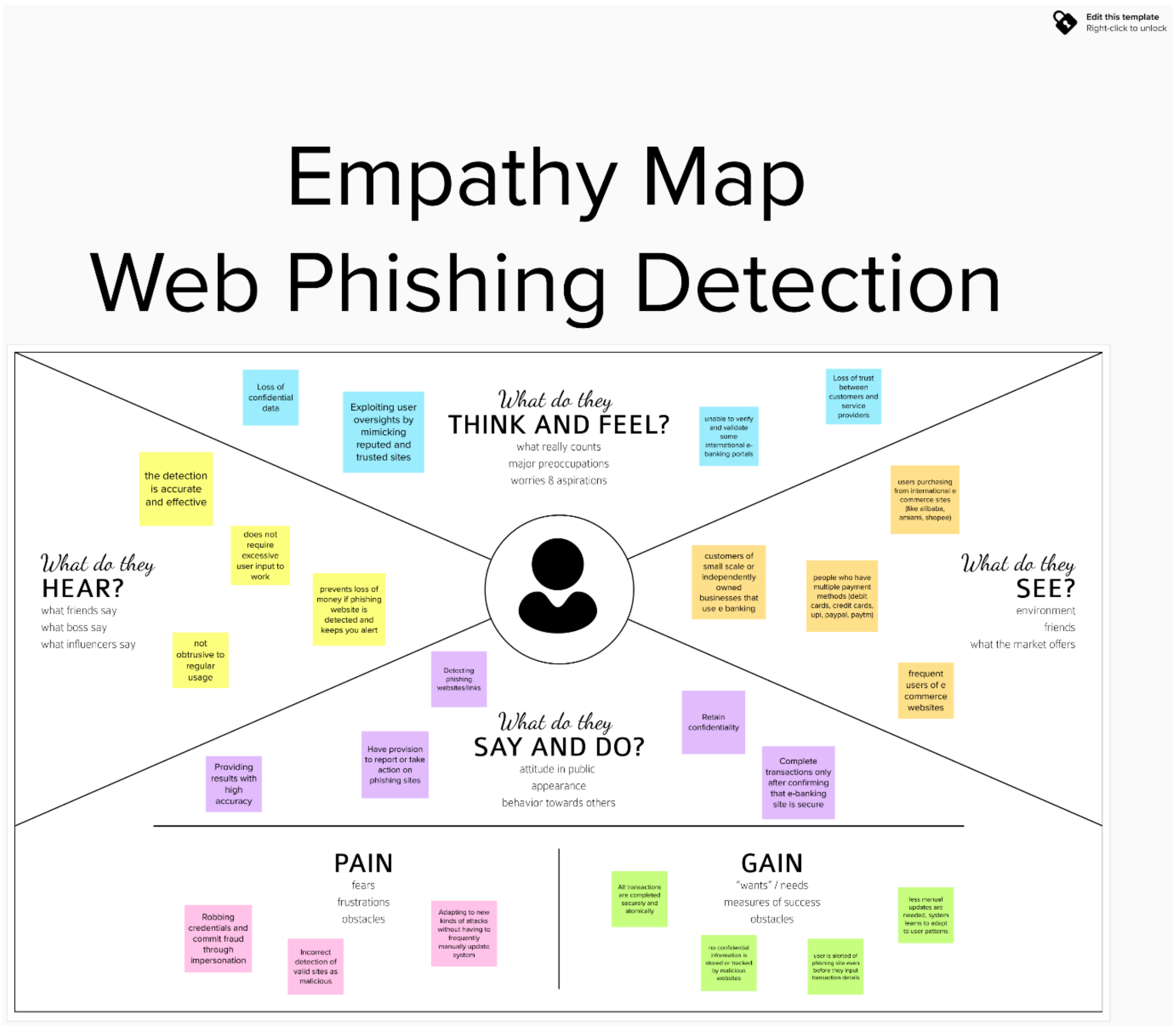
2.3 PROBLEM STATEMENT DEFINITION

The goal of this project is to create a model that will be able to recognize and classify whether a specific website is a phishing website or not by extracting and analysing the relevant features from the website's URL. The major goal of this system is to understand the working Random Forest Machine Learning Model and apply it to the web phishing detection system.

3. IDEATION AND PROPOSED SOLUTION

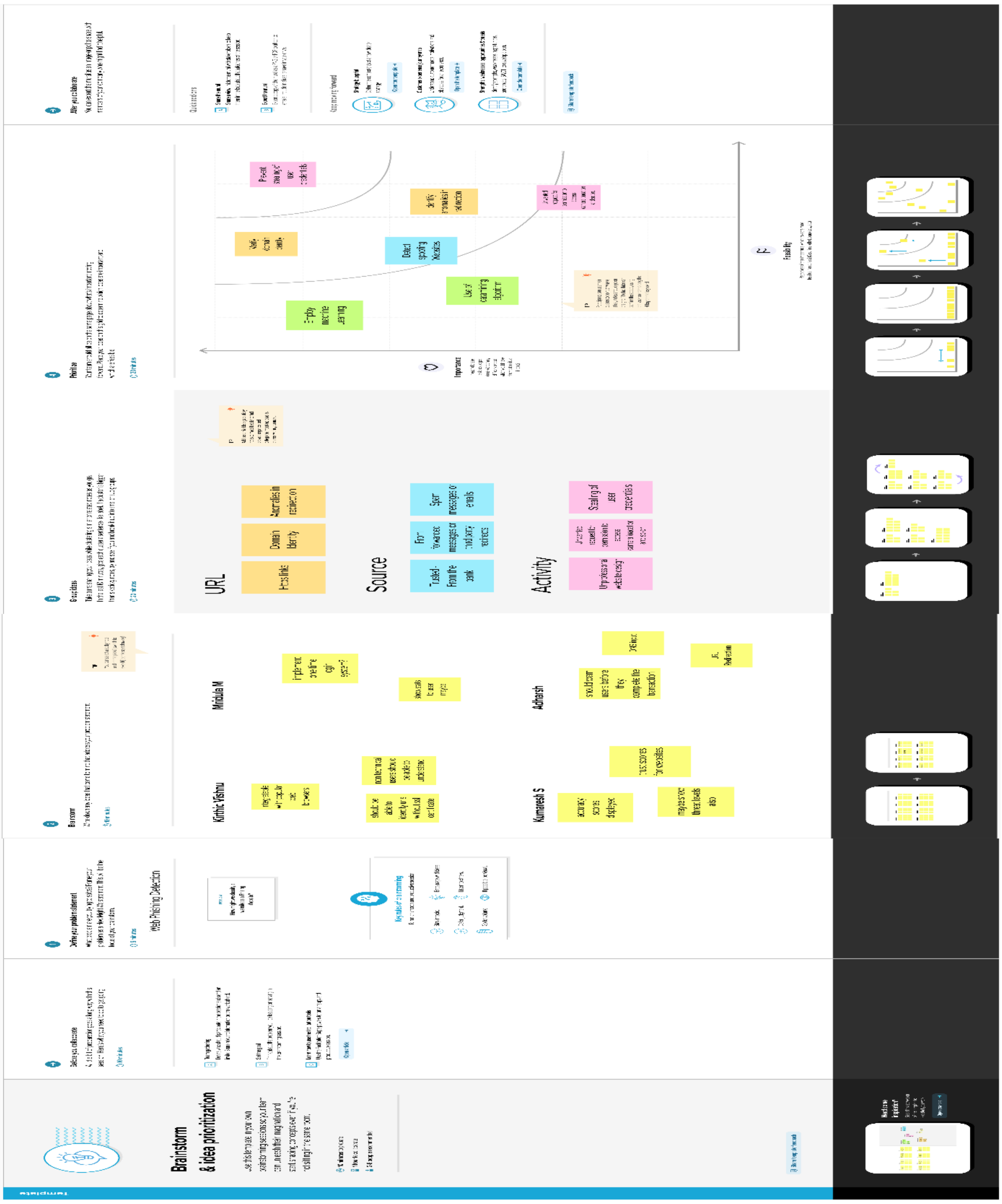
3.1 EMPATHY MAP CANVAS

An empathy map is a collaborative tool that teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.


Edit this template
Right-click to unlock
Empathy Map
Web Phishing Detection
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Loss of confidential data
Exploiting user oversights by mimicking reputed and trusted sites
unable to verify and validate some international e-banking portals
Loss of trust between customers and service providers
What do they HEAR?
what friends say
what boss say
what influencers say
the detection is accurate and effective
does not require excessive user input to work
prevents loss of money if phishing website is detected and keeps you alert
not obtrusive to regular usage
What do they SEE?
environment
friends
what the market offers
users purchasing from international e commerce sites (like alibaba, amiami, shopee)
customers of small scale or independently owned businesses that use e banking
people who have multiple payment methods (debit cards, credit cards, upi, paypal, paytm)
frequent users of e commerce websites
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Detecting phishing websites/links
Have provision to report or take action on phishing sites
Providing results with high accuracy
Retain confidentiality
Complete transactions only after confirming that e-banking site is secure
PAIN
fears
frustrations
obstacles
Robbing credentials and commit fraud through impersonation
Incorrect detection of valid sites as malicious
Adapting to new kinds of attacks without having to frequently manually update system
GAIN
"wants" / needs
measures of success
obstacles
All transactions are completed securely and atomically
no confidential information is stored or tracked by malicious websites
user is alerted of phishing site even before they input transaction details
less manual updates are needed, system learns to adapt to user patterns

3.2 IDEATION AND BRAINSTORMING

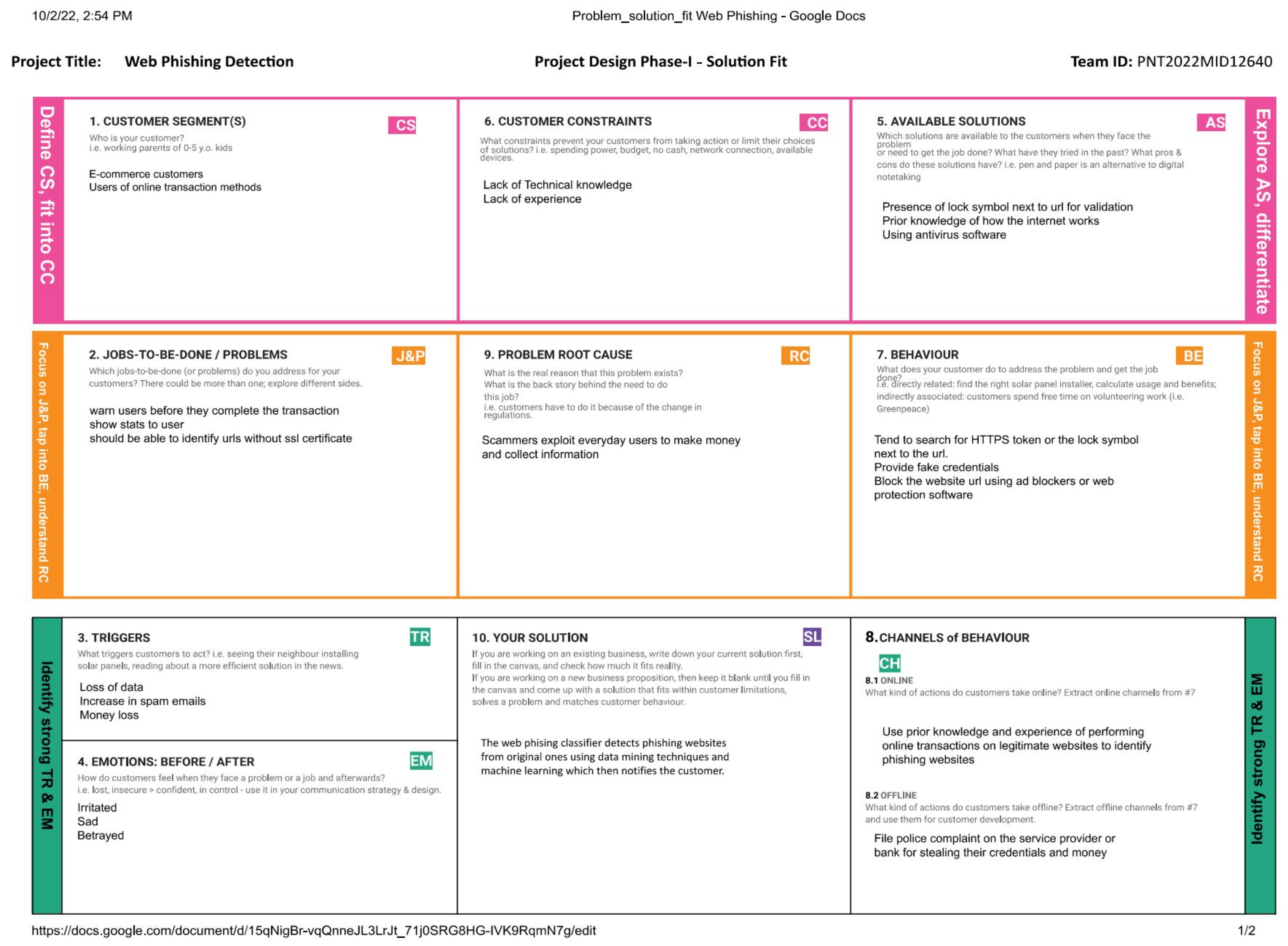
Ideation refers to the whole creative process of coming up with and communicating new ideas. Brainstorming is a collaborative problem-solving method that involves spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.



3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Many customers use e-banking to pay for their online purchases of goods. Some e-banking websites, for malicious content, request users to submit private information like usernames, passwords, credit card numbers, and more. Phishing websites are this kind of online banking website. One of the most vital software services for communications over the Internet is the web service. One of the many security risks to web services on the Internet is web phishing.
2.	Idea / Solution description	By extracting pertinent features from the target website, conducting feature co-relation, and then passing the information through the C5 classifier, the proposed method aids the user in distinguishing a legitimate website from a malicious one. Using the results obtained safe websites for online transactions are acquired which would then be made accessible to the users through an online application.
3.	Novelty / Uniqueness	The proposed solution uses novel methods to identify relevant features by factoring in attributes like page rank, in-degree, presence of HTTPS token, and others. Through the features obtained, different dataset sizes and distributions are tested for analyzing and detecting optimal hyperparameters.
4.	Social Impact / Customer Satisfaction	The proposed method performs better than numerous other traditional classification algorithms since it uses data mining algorithms. Inherently, users can also buy things online with the aid of this technique without any reluctance or second thoughts about security concerns.
5.	Business Model (Revenue Model)	Micro web frameworks like flask can be used to create a REST-based web application that users may use to conduct reliable and secure online transactions through safe e-commerce websites. Based on membership levels, different levels of security strictness and multiple volumes of secure e-commerce websites would be offered.
6.	Scalability of the Solution	The proposed website's functionality can be turned into an API so that other e-banking and e-commerce portals can utilize it to provide their users with safer and more effective solutions. It could be further extended for other security concerns such as audio recording, video recording, location tracking, virus attacks, and more, with safer and more effective solutions.

3.4 PROBLEM SOLUTION FIT



10/2/22, 2:54 PM Problem_solution_fit Web Phishing - Google Docs

Project Title: **Web Phishing Detection** **Project Design Phase-I – Solution Fit** **Team ID:** PNT2022MID12640

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

E-commerce customers
Users of online transaction methods

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Lack of Technical knowledge
Lack of experience

5. AVAILABLE SOLUTIONS AS

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

Presence of lock symbol next to url for validation
Prior knowledge of how the internet works
Using antivirus software

Explore AS, differentiate

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

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

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

warn users before they complete the transaction
show stats to user
should be able to identify urls without ssl certificate

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Scammers exploit everyday users to make money and collect information

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Tend to search for HTTPS token or the lock symbol next to the url.
Provide fake credentials
Block the website url using ad blockers or web protection software

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

Identify strong TR & EM

3. TRIGGERS TR

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

Loss of data
Increase in spam emails
Money loss

4. EMOTIONS: BEFORE / AFTER EM

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

Irritated
Sad
Betrayed

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

The web phising classifier detects phishing websites from original ones using data mining techniques and machine learning which then notifies the customer.

8. CHANNELS of BEHAVIOUR

CH

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

Use prior knowledge and experience of performing online transactions on legitimate websites to identify phishing websites

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

File police complaint on the service provider or bank for stealing their credentials and money

Identify strong TR & EM

https://docs.google.com/document/d/15qNigBr-vqQnneJL3LrJt_71j0SRG8HG-IVK9RqmN7g/edit 1/2

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Website Form and other platforms.
FR-2	User Confirmation	Confirmation through Email.
FR-3	User Authentication	Authentication using a password.
FR-4	User Input	User enters the URL they wish to check credibility for.
FR-5	Website Comparison	Machine Learning model using the URL entered by the user to compare with other URLs using Blacklist and Whitelist concepts.
FR-6	Feature Extraction	Features of the website such as visuals and contents of the website are extracted based on heuristics.
FR-7	Prediction	Machine Learning model employs ML algorithms to predict whether the website is malicious or not.
FR-8	Classifier	The prediction model sends its output to the classifier model and produces the final result.
FR-9	Announcement	Model provides the probability score of the site being legal compared to the site being a phishing website.

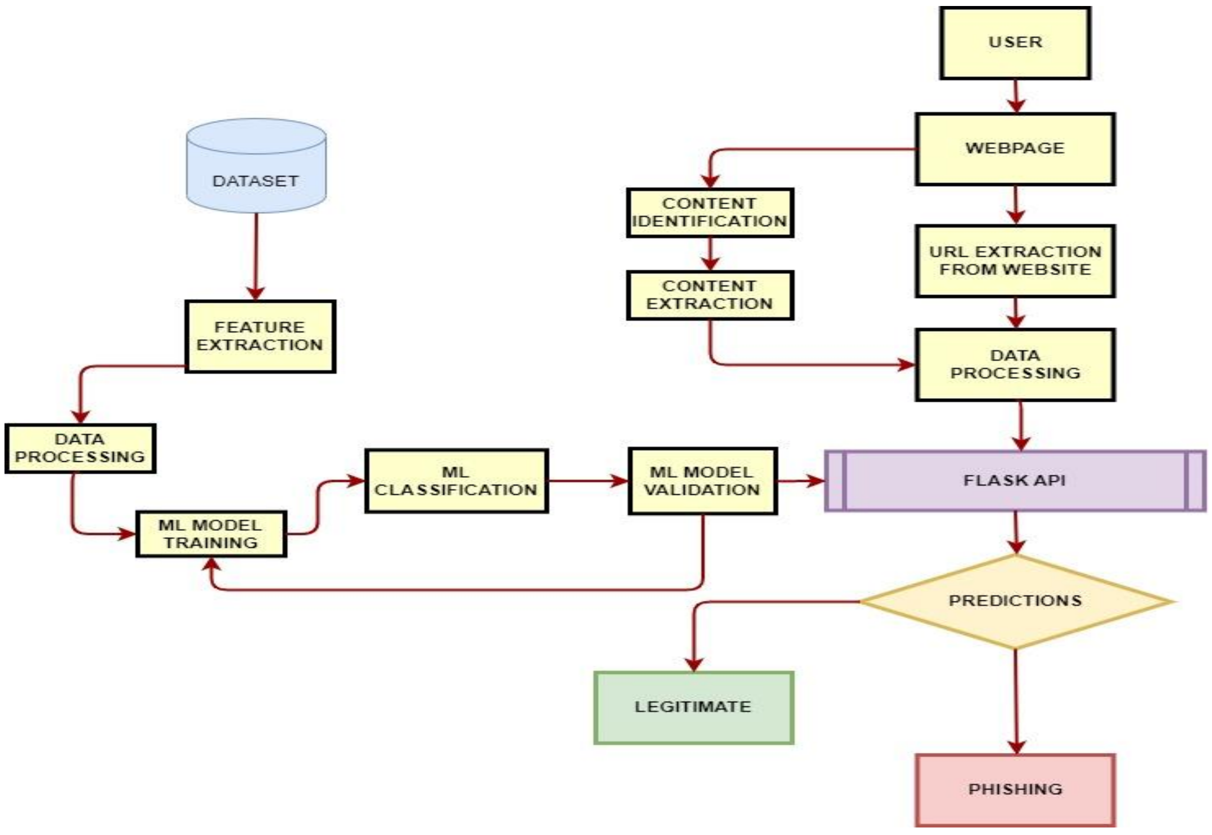
4.2 NON-FUNCTIONAL REQUIREMENT

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

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The system is usable and flexible to any of the user’s input meaning different forms of URLs.
NFR-2	Security	No user data will be leaked and the system is protected against any malware attacks or unauthorized access.
NFR-3	Reliability	The system is comparable to manual URL classification.
NFR-4	Performance	Parameters for performance measurement are done through a confusion matrix, Matthews correlation coefficient, and ROC AUC score.
NFR-5	Availability	The system will be accessible to the user at any point in time through a web browser.
NFR-6	Scalability	The design of the pipeline would be able to handle erratic demands with great efficiency.

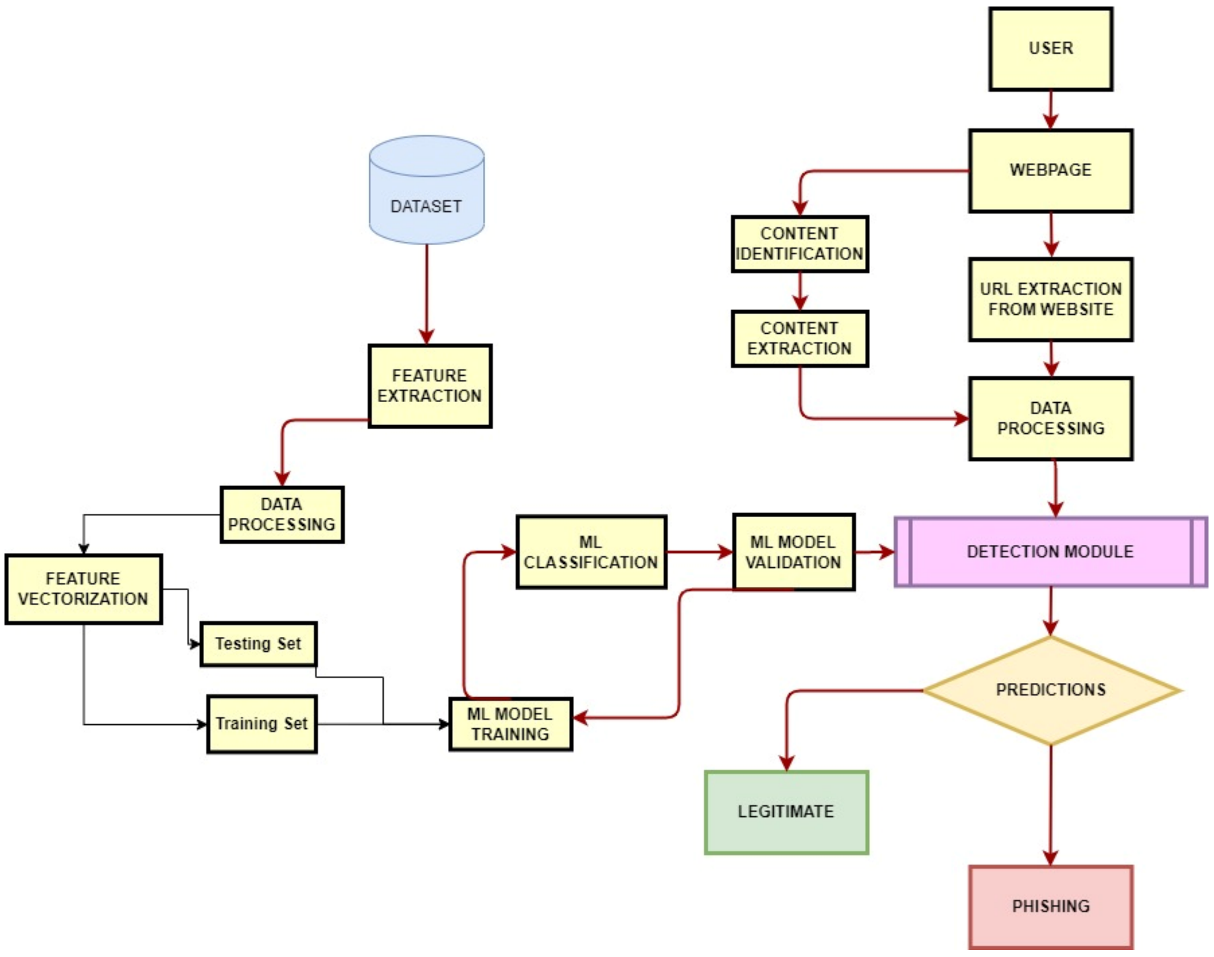
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAM

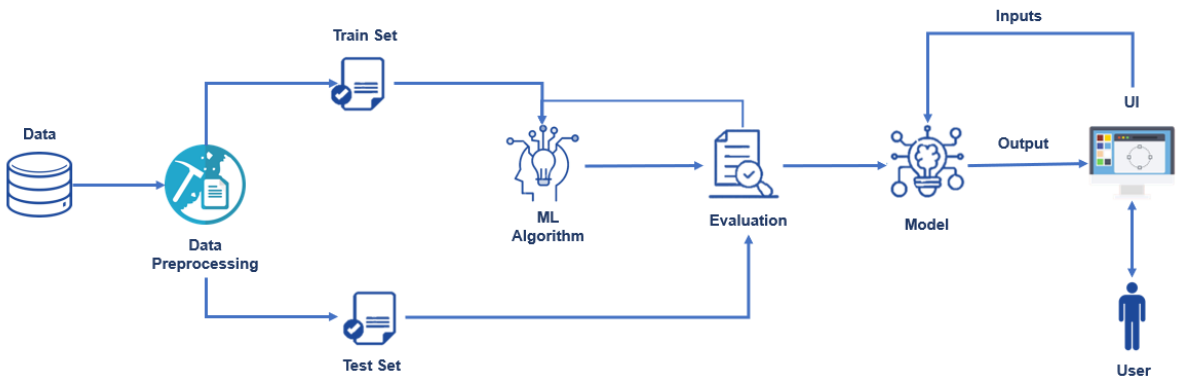


5.2 TECHNICAL ARCHITECTURE

5.2.1 SOLUTION ARCHITECTURE



5.2.2 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, and password, and confirming my password.	I can access my account dashboard	High	Sprint-1
		USN-2	As a user, I will receive a confirmation email once I have registered for the application	I can receive a confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through other platforms.	I can register & access the dashboard through other login platforms	Low	Sprint-2
	Login	USN-5	As a user, I can log into the application by entering my email & password	I clog ingin using my respective credentials	High	Sprint-1
	Dashboard	USN-6	As a user, I can navigate the intuitive dashboard to complete the task	Intuitive and easy-to-use dashboard	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Login & Dashboard	USN-7	As a user, I can navigate the application as I did use my mobile. I can access the same resources	I can log in to the application and access the resources using the dashboard	High	Sprint-1
Customer Care Executive	Login	CCE-1	As a CCE, I can log in to the application using the respective credentials and I can interact with the users	I can log in using my credentials	High	Sprint-1
	Dashboard	CCE-2	As a CCE, I can view all user queries and respond appropriately	The dashboard displays all the queries and offers response capabilities	High	Sprint-2
Administrator	Login & Dashboard	A-1	As an admin, I can log in and manage the activities	I can log in and interact with the application's features	High	Sprint-1

6. PROJECT DESIGN

6.1 SPRINT PLANNING AND ESTIMATION

Title	**Description**	**Date**
Literature Survey and Information Gathering	Gathering information by referring to technical papers, research, research publication, etc,.	20 OCTOBER 2022
Prepare Empathy Map	To capture user's pains and gains. Prepare a list of problem statements	20 OCTOBER 2022
Ideation	Prioritize the top 3 ideas based on feasibility and importance	20 OCTOBER 2022
Proposed Solution	Solution includes novelty, feasibility, business model, social impact, and scalability of the solution	20 OCTOBER 2022
Problem Solution Fit	Solution Fit Document	20 OCTOBER 2022
Solution Architecture	Solution Architecture	20 OCTOBER 2022
Customer Journey	To understand user interactions and experiences with the application	21 OCTOBER 2022
Functional Requirement	Prepare functional requirements documents	21 OCTOBER 2022
Data Flow Diagram	Data Flow Diagram	21 OCTOBER 2022
Technology Architecture	Technology Architecture Diagram	21 OCTOBER 2022
Milestone and Sprint Delivery Plan	Activities that have been performed and further plans	31 OCTOBER 2022
Project Development Delivery of Spirit 1,2,3 & 4	Develop and submit the proposed solution by testing it	26 OCTOBER 2022 – 19 NOVEMBER 2022

6.2 SPRINT DELIVERY SCHEDULE

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, and password, and confirming my password.	2	High	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-1	Login	USN-2	As a user, I will receive a confirmation email once I have registered for the application	1	High	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-2	Upload URL	USN-3	As a user, I can upload an URL to the application	2	Medium	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-2	Prediction	USN-4	As a user, I can predict wether the URL is malicious or not	`1	Medium	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-3	Upload URL	USN-5	As a user, I can upload an URL to the application	2	High	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-3	Choice to continue	USN-6	As a user, I can choose to continue to the malicious website	1	Medium	A.Kirthic Adharsh.S Kumaresh.S Mridula.M
Sprint-4	Classify as malicious or not	USN-7	As a user, I can predict whether the URL is malicious or not	1	Medium	A.Kirthic Adharsh.S Kumaresh.S Mridula.M

7. CODING AND SOLUTIONS

7.1 FEATURE 1

Using Random Forest Model in our Project:

Random forest is a tree-based algorithm that involves building several trees (decision trees), then combining their output to improve the generalization ability of the model. The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone.

7.2 FEATURE 2

Using Flask application in our Project:

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools

8. TESTING

8.1 TEST CASES

Testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps, or missing requirements in contrary to the actual requirement

i) Unit Testing:

When the testing happens for some individual group or some related units then that type of testing is called Unit Testing. It is often done by a programmer to test the part of the program he or she has implemented. Unit Testing is successful means all the modules have been successfully tested and it can proceed further.

ii) Functional Testing:

This type of testing is tested to check the functional components, or the functionality required from the system is gained or not. It falls under the testing of Black Box testing of Software Engineering. This part includes the feeding of the inputs in the system or the project and checking if that system or the project is getting the same value or not as expected if not then calculate the error as wanted and check for more. Functional Testing of this project mainly involves below things. All of these are tested successfully, and errors are also calculated.

a) Verifying the input image ii)Verifying the workflow
b) Correct recognition and calculate the error

iii) Integration Testing:

In a total project or system, many groups of components are getting added or summed up for the purpose of the project query. Integration testing is about checking the interaction between various modules of the project or the system. This module also includes the hardware and software requirements of the project. All the individual modules are integrated and tested together. All the best and extreme cases that the modules are interacting or not are successfully checked and passed, and errors are calculated for the deep learning platforms.

iv) System Testing:

This type of testing is actually meant for the system or the project and also the platform and the integrated software and tools, technologies are also tested. The idea or purpose behind the system testing is to check all the requirements that will be provided by the system. This application of the project along with the tools and technologies has been tested in both windows and Linux. It passed successfully.

8.2 USER ACCEPTANCE TESTING

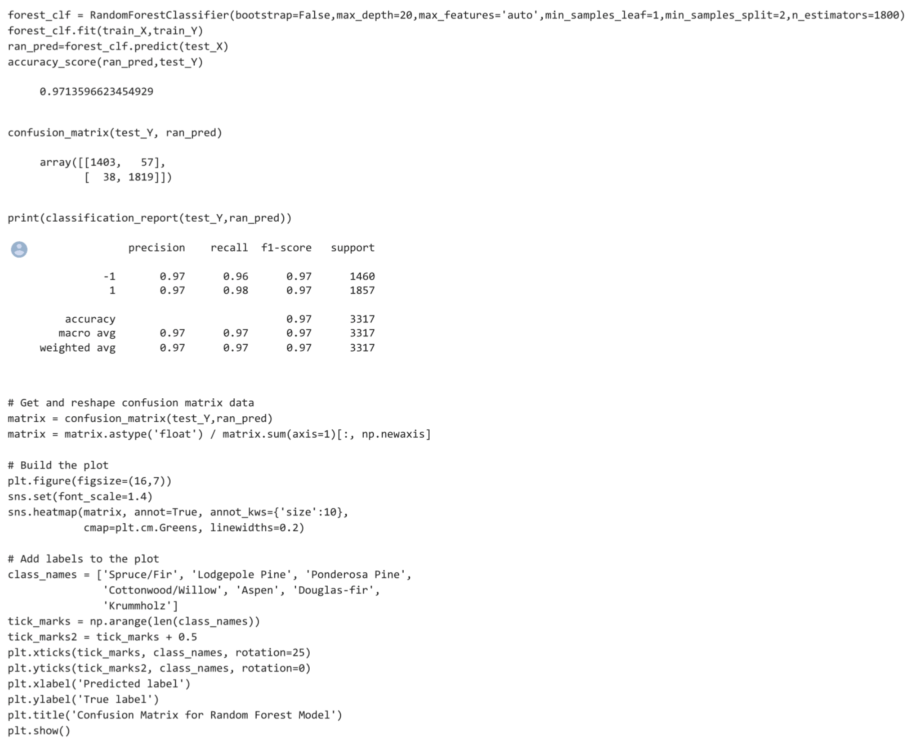
This is a type of system or software testing where a system has been tested for availability. The purpose of this test is to check the business requirements and assess whether it will be accepted for delivery.

9. RESULTS

9.1 PERFORMANCE METRICS

9.1.1 MODEL METRICS

Our model performs with 97% of accuracy when training and testing sessions.



```
forest_clf = RandomForestClassifier(bootstrap=False,max_depth=20,max_features='auto',min_samples_leaf=1,min_samples_split=2,n_estimators=1800)
forest_clf.fit(train_X,train_Y)
ran_pred=forest_clf.predict(test_X)
accuracy_score(ran_pred,test_Y)

     0.9713596623454929


confusion_matrix(test_Y, ran_pred)

     array([[1403,   57],
            [  38, 1819]])


print(classification_report(test_Y,ran_pred))

                   precision    recall  f1-score   support

              -1       0.97      0.96      0.97      1460
               1       0.97      0.98      0.97      1857

        accuracy                           0.97      3317
       macro avg       0.97      0.97      0.97      3317
    weighted avg       0.97      0.97      0.97      3317


# Get and reshape confusion matrix data
matrix = confusion_matrix(test_Y,ran_pred)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Build the plot
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Add labels to the plot
class_names = ['Spruce/Fir', 'Lodgepole Pine', 'Ponderosa Pine',
               'Cottonwood/Willow', 'Aspen', 'Douglas-fir',
               'Krummholz']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=25)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
plt.show()
```

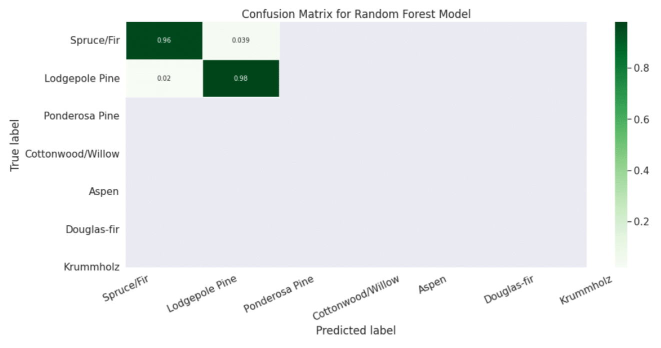
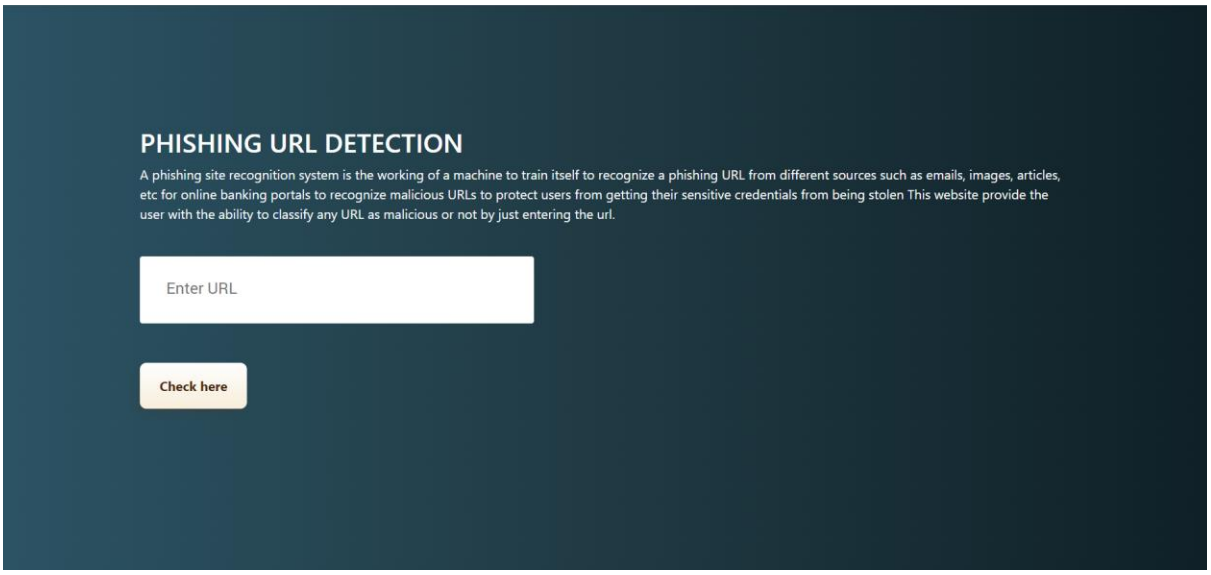


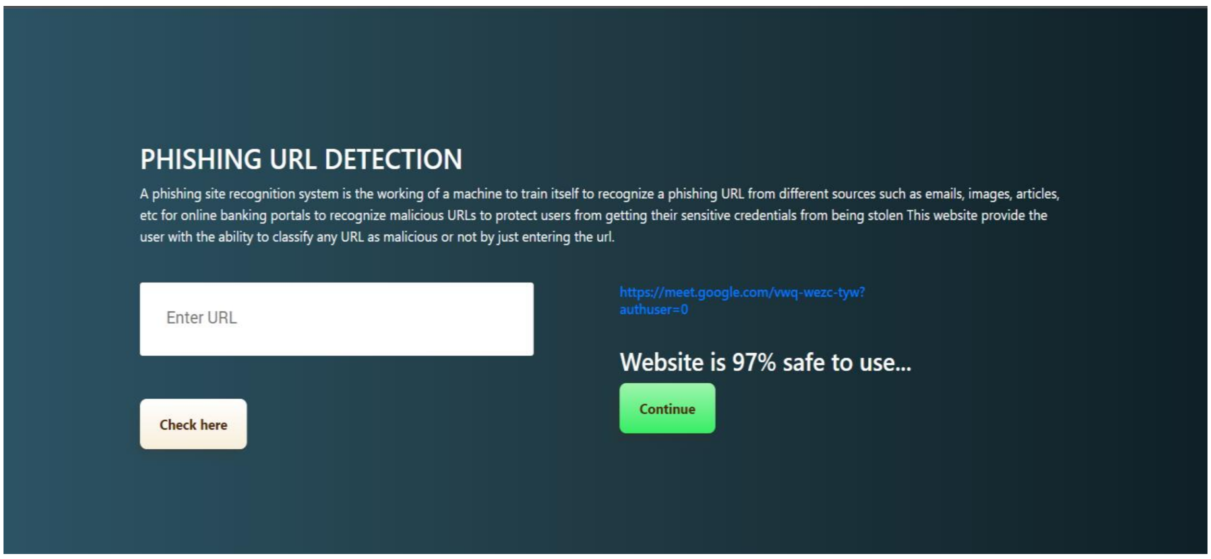
Fig 1 – Screenshot of Model MEtrics

10. OUTPUT SCREENSHOTS


PHISHING URL DETECTION

A phishing site recognition system is the working of a machine to train itself to recognize a phishing URL from different sources such as emails, images, articles, etc for online banking portals to recognize malicious URLs to protect users from getting their sensitive credentials from being stolen This website provide the user with the ability to classify any URL as malicious or not by just entering the url.

Fig 2 – Screenshot of the web application

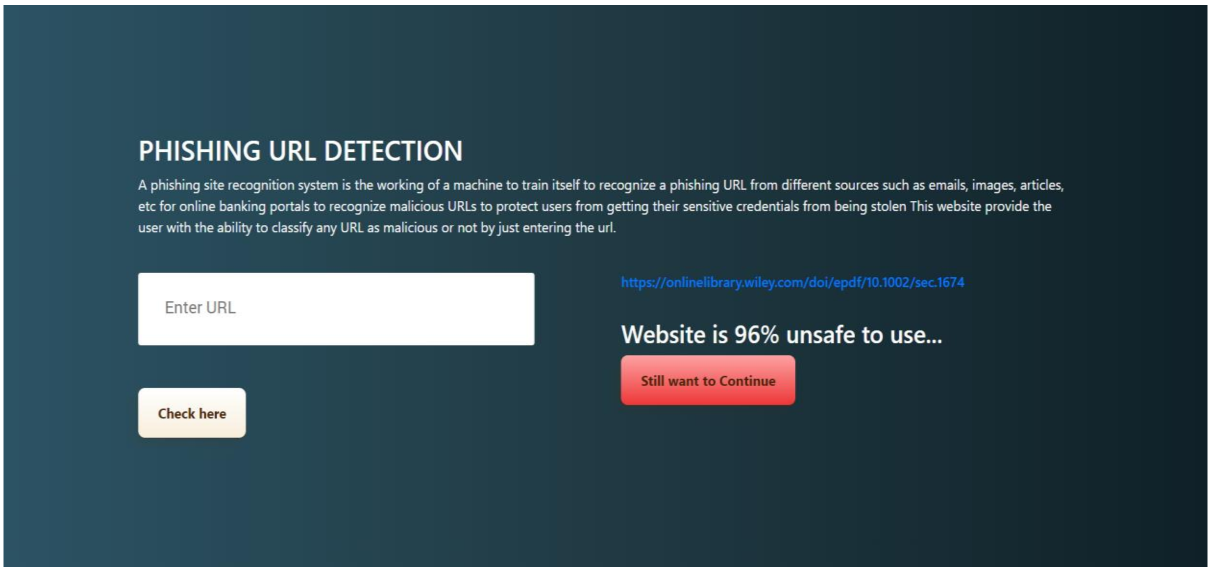

PHISHING URL DETECTION

A phishing site recognition system is the working of a machine to train itself to recognize a phishing URL from different sources such as emails, images, articles, etc for online banking portals to recognize malicious URLs to protect users from getting their sensitive credentials from being stolen This website provide the user with the ability to classify any URL as malicious or not by just entering the url.

https://meet.google.com/vwq-wezc-tyw?authuser=0

Website is 97% safe to use...

Fig 3 – Screenshot of the classifying a URL as safe


PHISHING URL DETECTION

A phishing site recognition system is the working of a machine to train itself to recognize a phishing URL from different sources such as emails, images, articles, etc for online banking portals to recognize malicious URLs to protect users from getting their sensitive credentials from being stolen This website provide the user with the ability to classify any URL as malicious or not by just entering the url.

https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.1674

Website is 96% unsafe to use...

Fig 4 – Screenshot of the classifying a URL as safe

11. ADVANTAGES AND DISADVANTAGES

11.1 ADVANTAGES

1) The system not only produces a classification of the phishing URLs but also a rich description of the features contributing to the co-relation mapping
2) The method involves a relatively small number of parameters and hence training is relatively easy and fast.
3) The use of the Random Forest Model helps with identifying the necessary co-relation between features that help classify the URL.

11.2 DISADVANTAGES

1) It can only classify malicious links which are direct links to malicious websites
2) Cannot extract latent features from the URL that characterizes certain URLs as malicious
3) Cannot check for batch inputs

12. CONCLUSION

Web Phishing Detection using Deep Learning methods has been implemented. Random Forest Model have been trained and tested on the same data in order to acquire the comparison between the classifiers. Utilizing these deep learning techniques, a high amount of accuracy can be obtained.

13. FUTURE SCOPE

The proposed system cannot extract the latent features from certain malicious URLs hence classifying them as safe, future works can include identifying these latent features and extracting them for the model to learn to provide more accurate predictions.

14. APPENDIX

Python:

Python is an interpreted, high-level, general-purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant White space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed, and garbage is collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Keras:

Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. It uses libraries such as Python, C#, C++, or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks

Numpy:

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. Numpy which stands for Numerical Python is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray

Machine Learning:

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns, and make decisions with minimal human intervention.

Deep Learning:

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision-making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network.

Source Code

Index.html

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

   <!-- BootStrap -->
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">

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

</head>

<body>

<div class=" container">
   <h2>PHISHING URL DETECTION</h2>
   <p>A phishing site recognition system is the working of a machine to train itself to recognize a
phishing URL from different sources such as emails, images, articles, etc for online banking
portals to recognize malicious URLs to protect users from getting their sensitive credentials from
being stolen

      This website provide the user with the ability to classify any URL as malicious or not by just
entering the url.
   </p>
   <div class="row">
      <div class="form col-md" id="form1">


         <br>
         <form action="/" method ="post">
            <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL"
required="" />
```

```
            <label for="url" class="form__label">URL</label>
            <button class="button" role="button" >Check here</button>
          </form>


      </div>

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

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

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

      <!-- JavaScript -->
      <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+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}}';
            let num = x*100;
            console.log(num,x);
            if (0<=x && x<0.50){
                num = 100-num;
            }
            console.log(num,x);
            let txtx = num.toString();
            if(x<=1 && x>=0.60){
                var label = "Website is "+txtx +"% safe to use...";
                document.getElementById("prediction").innerHTML = label;
                document.getElementById("button1").style.display="block";
            }
            else if (0<=x && x<0.60){
                var label = "Website is "+txtx +"% unsafe to use..."
                document.getElementById("prediction").innerHTML = label ;
                document.getElementById("button2").style.display="block";
            }


    </script>

</body>

</html>
```

feature.py

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

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

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

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

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




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


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

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

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


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

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

    # 3.shortUrl
    def shortUrl(self):
        match = 
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                    'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                    'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.u
s|'
                    '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.Symbol@
    def symbol(self):
        if re.findall("@",self.url):
            return -1
        return 1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
```

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

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

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

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

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

# 25. DNSRecording
def DNSRecording(self):
    try:
        creation_date = self.whois_response.creation_date
```

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

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

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

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

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


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

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

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(
        'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.l
t|ow\.ly', self.url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\
.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'
                                '107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.
151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.2
25|'
                                '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.12
6\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.15
3|'
                                '216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.
170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.12
7\.157|'
                                '34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.14
7\.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\.22
5\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1
    def getFeaturesList(self):
        return self.features
```

app.py

```
from flask import Flask,render_template,request,jsonify
import numpy as np
import pickle
from feature import FeatureExtraction

app = Flask(__name__)
#Load model
model = pickle.load(open('web_phishing_model.bin','rb'))


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

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

        y_pred =model.predict(x)[0]
        print(y_pred)
        #1 is safe
        #-1 is unsafe

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

if __name__ == "__main__":
    app.run(debug=True)
```