WEB PHISHING DETECTION

NALAIYA THIRAN PROJECT BASED LEARNING

On

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY, AND ENTREPRENEURSHIP

A PROJECT REPORT TEAM ID: PNT2022TMID46988

Team members:

SANTHANAKUMAR V	-	822119104034
KRISHNARAJ R	-	822119104020
VARATHARAJ	-	822119104045
VASANTHAKUMAR	-	822119104046

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING - PATTUKKOTTAI

(A Constituent College of Anna University, Chennai)

PATTUKKOTTAI – 614701

NOVEMBER 2022

ABSTRACT

Phishing is the most commonly used social engineering and cyber attack. Through such attacks, the phisher targets naive online users by tricking them into revealing confidential information, with the purpose of using it fraudulently. In order to avoid getting phished, Users should have awareness of phishing websites. Have a blacklist of phishing websites which requires the knowledge of website being detected as phishing. Detect them in their early appearance, using machine learning and deep neural network algorithms. Of the above three, the machine learning based method is proven to be most effective than the other methods. A phishing website is a common social engineering method that mimics trustful uniform resource locators (URLs) and webpages. The objective of this project is to train machine learning models and deep neural nets on the dataset created to predict phishing websites. Both phishing and benign URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted. The performance level of each model is measured and compared. Keywords: Deep learning, Machine learning, Phishing website attack, Phishing website detection, Anti-phishing website, Legitimate website , Phishing website datasets, Phishing website features.

PRE-REQUISITES TOOLS : JUPITER NOTEBOOK

OPERATING SYSTEM : WINDOWS 10

LANGUAGE : PYTHON

INSTALLING LIBRARIES

In this first step, we have to import the most common libraries used in python for machine learning such as

- Pandas
- Numpy
- Seaborn
- Matplotlib

IMPORTING DATA

In this project, we have used the url pre processed data.

CHAPTER 1

INTRODUCTION

Phishing imitates the characteristics and alternatives of emails and makes it appear similar due to the fact the original one. It seems nearly like that of the legitimate supply. The consumer thinks that this e-mail has come back from a real employer or a corporation. This makes the consumer to forcefully visit the phishing internet site thru the hyperlinks given inside the phishing email. These phishing web sites region unit created to mock the seams of an ingenious website. The phishers force person to inventory up the non-public info via giving baleful messages or validate account messages etc. so that they inventory up the preferred data which might be utilized by them to misuse it. They devise things such as the user isn't always left with the other choice but to go to their spoofed web site. Phishing is the most hazardous criminal physical activities in the cyber region. Since the maximum of the customers logs on to get admission to the services supplied with the aid of government and financial establishments, there has been a significant boom in phishing attacks for the beyond few years. Phishers commenced to earn cash and that they try this as a thriving business.

Phishing may be law-breaking, the explanation behind the phishers doing this crime is that it is terribly trustworthy to try to do this, it doesn't value something and it effective. The phishing will truly get entry to the e-mail identity of somebody it's terribly sincere to are looking for out the email identification currently every day and you will send an email to every person is freely offered throughout the globe. These attacker's vicinity terribly much less price and electricity to urge valuable know-how quick and truly. The phishing frauds effects malware infections, statistics loss, fraud, etc. information at some stage in which those cyber criminals have an interest is that the crucial data of a user similar to the password, OTP, credit/ debit card numbers CVV, sensitive know- how associated with business, medical understanding, confidential information, etc commonly these criminals conjointly acquire data which may provide them directly get admission to do the social media account their emails. 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.

1.1 PROJECT OVERVIEW

- To develop a novel approach to detect malicious URL and alert users.
- To apply ML techniques in the proposed approach in order to analyze the real time URLs and produce effective results.
- To implement the concept of RNN, which is a familiar ML technique that has the capability to handle huge amount of data.

1.2 PURPOSE

- To develop an unsupervised deep learning method to generate insight from a URL.
- The study can be extended in order to generate an outcome for a larger network and protect the privacy of an individual.

CHAPTER 2

LITERATURE SURVEY

Abstract:

Phishing is a common attack against Internet users that causes them to reveal their information using fake websites. The goal of the fake website is to steal personal information such as usernames, passwords and online banking transactions. Scammers use websites that are visually and semantically similar to the real ones.

As technology continues to advance, phishing techniques begin to advance rapidly, and this should be prevented by using anti-phishing mechanisms such as spoofed URL detection. Machine Learning is a powerful tool used to combat spoofing attacks. This report covers machine learning technology to detect fake URLs by extracting and analyzing different characteristics of legitimate and fake URLs. Random Forest, Logistic Regression and algorithms are used to detect fake websites.

Introduction:

Nowadays, the Internet plays an important role in communication, where people create an online environment to manage business functions, online activities of banks, social networks… However, the Internet also contains hidden things. A lot of risk because when users operate in an online environment they can be vulnerable to attackers. And their identity is often a fake URL. And spoofed URLs are often placed on popular websites or sent to user emails.

Literature Review:

Construction of Phishing Site. In the first step attacker identifies the target as a well-known organization. Afterward, attacker collects the detailed information about the organization by visiting their website. The attacker then uses this information to construct the fake website.

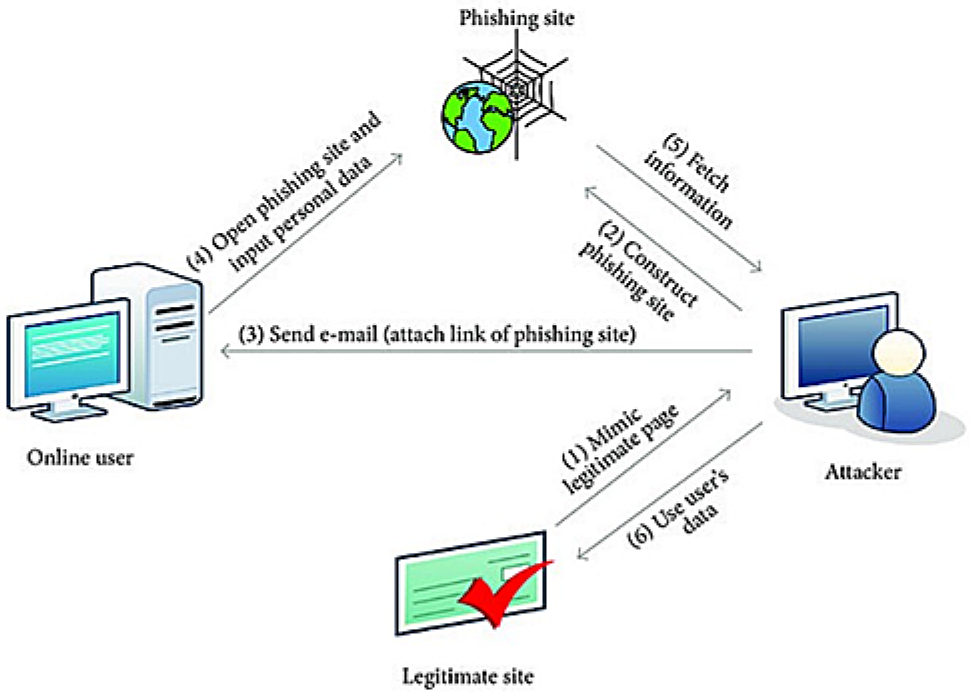
URL Sending. In this step, attacker composes a bogus e-mail and sends it to the thousands of users. Attacker attached the URL of the fake website in the bogus e-mail. In the case of spear phishing attack, an attacker sends the e-mail to selected users. An attacker can also spread the link of phishing website with the help of blogs, forum, and so forth [43].

Stealing of the Credentials. When user clicks on attached URL, consequently, fake site is opened in the web browser. The fake website contains a fake login form which is used to take the credential of an innocent user. Furthermore, attacker can access the information filled by the user.

Identity Theft. Attacker uses this credential of malicious purposes. For example, attacker purchases something by using credit card details of the user.

Although attacks use different techniques to create phishing websites to deceive users, most have similarly designed phishing website features. Therefore, researchers have conducted extensive anti-phishing research using phishing website features. Current methods for phishing detection include black and whitelists, heuristics, visual similarity, and machine learning, among which heuristics and machine learning are more widely used. The following is an introduction to

the aforementioned phishing detection techniques.
Black and whitelist
To prevent phishing attack threats, many anti-phishing methods have been proposed.
Blacklisting methods are the most straightforward ways to prevent phishing attacks and are widely used in the industry. Google Safe Browsing uses a blacklist-based phishing detection method to check if the URL of the matching website exists in the blacklist. If it does, it is considered a phishing website.


Phishing site
(4) Open phishing site and input personal data
(5) Fetch information
(2) Construct phishing site
(3) Send e-mail (attach link of phishing site)
Online user
Attacker
(1) Mimic legitimate page
(6) Use user's data
Legitimate site

FEATURES OF PROPOSED SYSTEM:

1) FUNCTIONAL CAPABILITIES: The ultimate aim of this project is to detect phishing attacks in real-time. This model checks the website with machine learning server for any maliciousness in the accessed site.
2) PERFORMANCE LEVEL: At the client side, it takes 1-2 seconds to detect whether a site is phishing or not.
3) DATA STRUCTURES: The data in this project are maintained in the CSV form. It provides easy access to the user.
4) SAFETY: No data loss occurs in this system
5) RELIABILITY: We assure that the project is completely authenticated in order to enhance security and corruptions of database as well as the software.

1.phishing detection and protection scheme :

Developing with the anti-phishing methods, phishers use various phishing methods and more complex and hard-to-detect approaches. The most straightforward way for a phisher to swindle people is to make the phishing web page similar to their target. However, many distinctive and features can distinguish the original legitimate website from the clone phishing website like the spelling error, image alteration, long URL address and abnormal DNS records.

The full list is revealed in Table 3 which is used later in our analysis and classification study. If an attacker clones a legitimate website as a whole or designed to look similar as they usually do in most attacks in recent times ,our approach is that similar looking phishing web page con-tent is not left for the users to check for the indicator or the authenticity attentively, but can detect by automated methods. Our approach is based on website phishing detection using the features of the site, content and their appearance. These properties are stored in a local database (Excel table) as a knowledge model and first compared with the newly loaded site at the time of loading against the dangerous web page offline. After the comparison was unable to detect the similarity, then the critical approach to compare the legitimate and fake using the features of the website with machine learning for an intelligent decision. The critical contribution of our approach includes Result .The output is determined by the classifier, in the phishing detection stage which predicts if the web page is suspicious, legiti-mate or phishing. The knowledge model and plug-in development will be developed at a later stage

2. System detection related work :

Nowadays most people uses internet for various purposes such as online shopping like purchasing or selling products, chat with friends, sending mail. Internet users now spend more time on social networking sites Information can spread very fast and easily within the social media networks. Social media systems depend on users for content contribution and sharing. Facebook had over 1.3 billion active users as of June 2014. There are over 1.3 billion (the number is keep growing) pages from various categories, such as company, product/service, musician/band, local business, politician, government, actor/director, artist, athlete, author, book, health, beauty, movie, cars, clothing, community. Fans not only can see information submitted by the page, but also can post comments, photos and videos to the page.

Result:

Domain anomaly features are used to identify possible malicious domains based on lexical and reputation factors, whereas social anomaly features represent anomalous user behaviors in social communications

3. Learning to Detect Phishing Emails :

An alternative for detecting these attacks is a relevant process of reliability of machine on a trait intended for the reflection of the besieged deception of user by means of electronic communication. This approach can be used in the detection of phishing websites, or the text messages sent through emails that are used for trapping the victims. Approximately, 800 phishing mails and 7,000 non-phishing mails are traced till date and are detected accurately over 95% of them along with the categorization on the basis of 0.09% of the genuine emails.

Result:

We can just wrap up with the methods for identifying the deception, along with the progressing nature of attacks.

4. Phishing websites machine learning:

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

Result:

Phishing URL detection plays a pivotal role for many cybersecurity software and applications. In this paper, we researched and reviewed works based on the advanced machine learning techniques and approaches that promise a fresh approach in this domain.

5. Support vector machine :

The existing anti-phishing approaches use the blacklist methods or features based machine learning techniques. Blacklist methods fail to detect new phishing attacks and produce high false positive rate. Moreover, existing machine learning based methods extract features from the third party, search engine, etc. Therefore, they are complicated, slow in nature, and not fit for the real-time environment. To solve this problem, this paper presents a machine learning based novel anti-phishing approach that extracts the features from client side only. Below architecture diagram as shown in Fig. 1. Represents mainly flow of training phase to Detection phase. First data need to be pre-processed and feature extraction using different feature sets and later we need to train this dataset with the corresponding algorithms and the output is displayed.

Result:

In future we can use a combination of any other two or more classifier to get maximum accuracy. We can also explore various phishing techniques that uses Lexical features.

QUALITY:

The project is developed with the help of Anaconda Navigator software which meets the requirement of the user, the project is checked whether the phases individually have a served its purpose.

REFERENCES:

[1] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor and J. Downs, “Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions”, Proceedings of the 28th international conference on Human factors in computing systems ser. CHI’10. New York NY USA:ACM, pp. 373-382, 2010.

[2] B. Krebs, “HBGary Federal HACKED by Anonymous”, December 2011,

[3] W. D. Yu, S. Nargundkar and N. Tiruthani, “A phishing vulnerability analysis of web based systems”, Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC 2008). Marrakech Morocco: IEEE, pp. 326-331, July 2008.

[4] P. Kumaraguru, Y. Rhee, A. Acquisti, L. F. Cranor, J. Hong and E. Nunge, “Protecting people from phishing: the design and evaluation of an embedded training email system”, Proceedings of the SIGCHI conference on Human factors in computing systems ser. CHI’07. New York NY USA: ACM, pp. 905-914, 2007.

[5] C. Yue and H. Wang, “Anti-phishing in offense and defense”, Computer Security Applications Conference 2008. ACSAC 2008. Annual, pp. 8-12, 2008.

[6] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong and C. Zhang, “An empirical analysis of phishing blacklists”, Proceedings of the 6th Conference in Email and Anti-Spam ser. CEAS’09 Mountain view CA, July 2009.

[7] Y. Zhang, J. I. Hong and L. F. Cranor, “Cantina: a content-based approach to detecting phishing web sites”, Proceedings of the 16th international conference on World Wide Web ser. WWW ‘07. New York NY USA:ACM, pp. 639-648, 2007.

[8] H. Zhang, G. Liu, T. Chow and W. Liu, “Textual and visual content-based anti-phishing: A -ayesian approach”, IEEE Transactions on Neural Networks, vol. 22, no. 10, pp. 1532-1546, oct. 2011.

2.3 PROBLEM STATEMENT DEFENETION

There are a number of users who purchase products online and make payments through ebanking. 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.

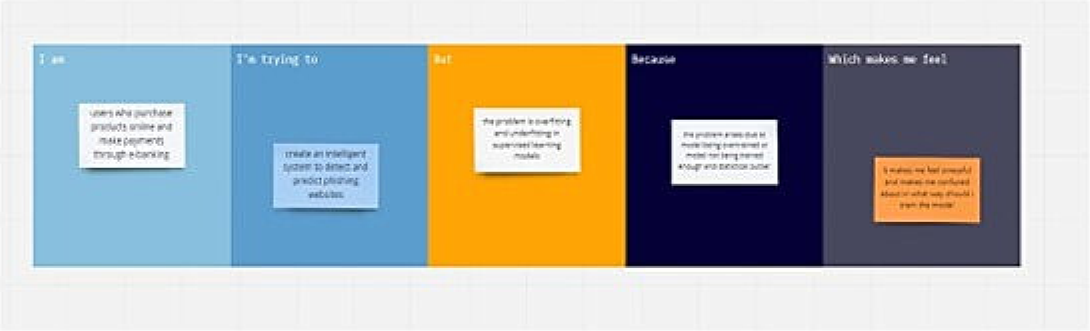
In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.

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.

Web phishing problem statement:

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makesme feel**
Web phishing detection	User who purchase products online and make payments through e-banking	Create an intelligent system to detect and predict phishing websites	Overfitting and underfitting of supervised learning models is an issue	Model being overtrained or model not being trained enough and statistical outliers	Stressful and confused

Web phishing detection problem statement:

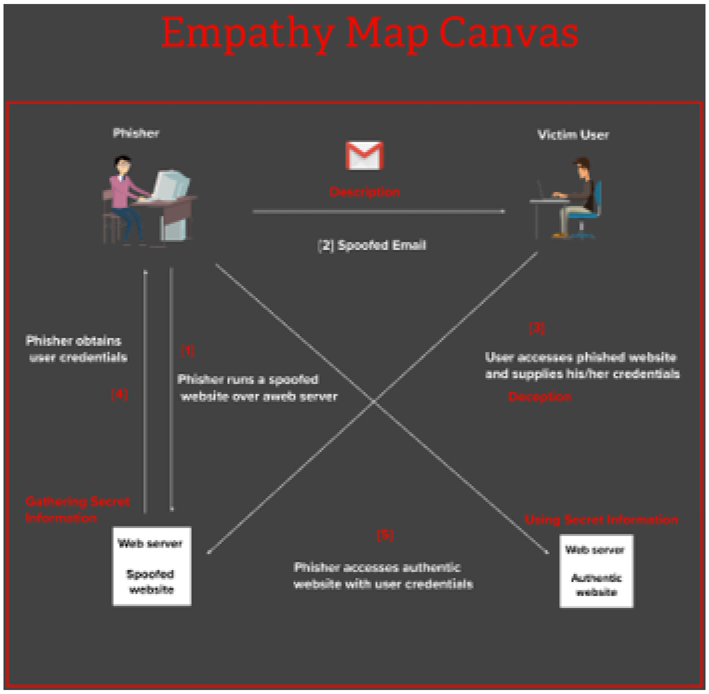


I am	I'm trying to	But	Because	Which makes me feel
users who purchase products online and make payments through e-banking	create an intelligent system to detect and predict phishing websites	the problem is overfitting and underfitting in supervised learning models	the problem arises due to model being overtrained or model not being trained enough and [illegible]	It makes me feel stressful and makes me confused about in what way should i train the model

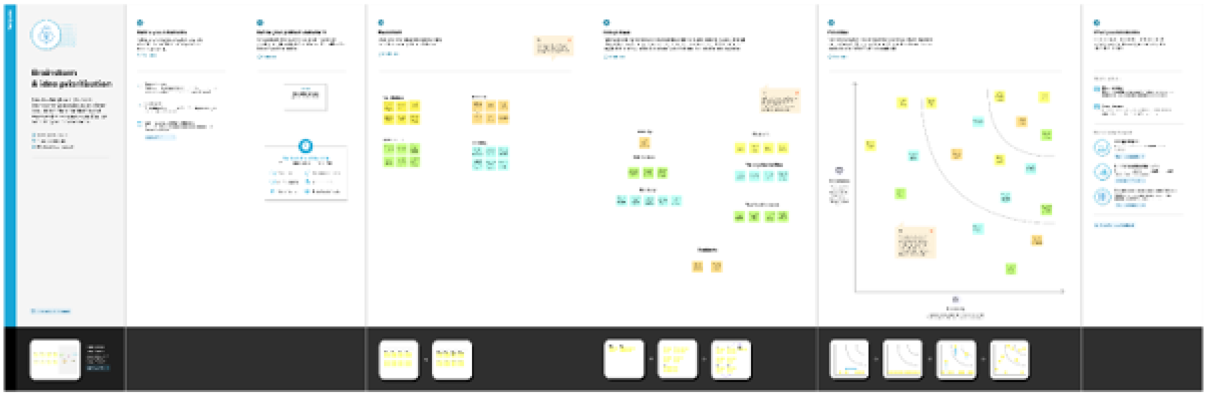
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


Empathy Map Canvas
Phisher
Victim User
Description
[2] Spoofed Email
Phisher obtains user credentials
[4]
[1]
Phisher runs a spoofed website over aweb server
[3]
User accesses phished website and supplies his/her credentials
Deception
Gathering Secret Information
Using Secret Information
Web server
Spoofed website
[5]
Phisher accesses authentic website with user credentials
Web server
Authentic website

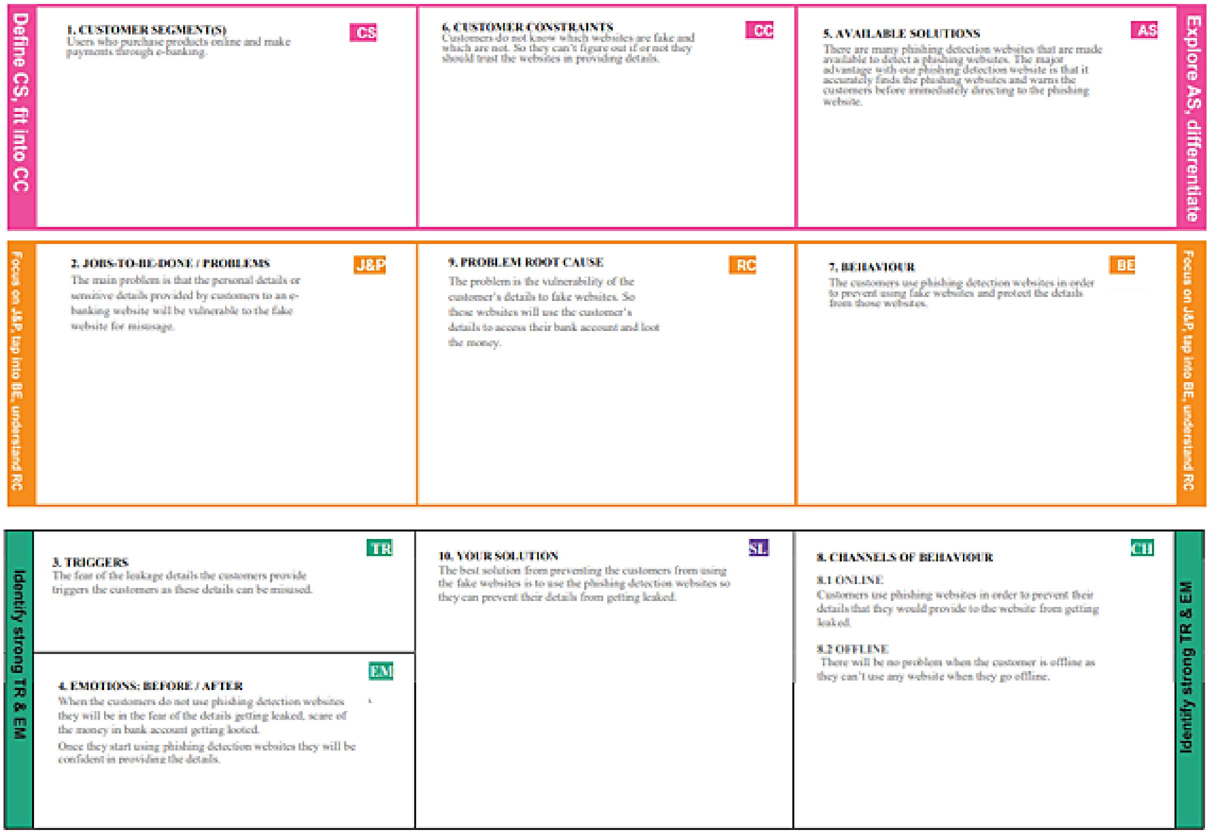
3.2 Ideation & Brainstorming



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	There are a number of users who purchase products onlineand 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 websiteisknown as a phishing website. Problem is to detect and predict e-banking phishing websites.
2.	Idea / Solution description	We have to build an intelligent system that will detect and predict phishing websites.
3.	Novelty / Uniqueness	It will detect the phishing websites accurately andnotify the users if itis a phishing website.
4.	Social Impact / Customer Satisfaction	Customers will get a notification that shows this page is not secure, the user is not reliable and do not open it or make any transactions. By warning the customer before opening thepage it makes the customer feel secure and help them detect phishing websites. So, the customers are highly satisfied.
5.	Business Model(Revenue Model)	This model gives high revenue because all theusers will use this web phishing detection because they don’t want to make unsafe transactions thatwill make them loose their money.
6.	Scalability of the Solution	The total execution time of our approach in phishing webpage detection is around 2–3 s, which is quite low and acceptable in a real-timeenvironment. As the input size increases execution time increases and this makes the system difficult to handle increasing stress.

3.4 Problem Solution fit:



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Users who purchase products online and make payments through e-banking.

6. CUSTOMER CONSTRAINTS CC

Customers do not know which websites are fake and which are not. So they can't figure out if or not they should trust the websites in providing details.

5. AVAILABLE SOLUTIONS AS

There are many phishing detection websites that are made available to detect a phishing websites. The major advantage with our phishing detection website is that it accurately finds the phishing websites and warns the customers before immediately directing to the phishing website.

Explore AS, differentiate

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

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

The main problem is that the personal details or sensitive details provided by customers to an e-banking website will be vulnerable to the fake website for misusage.

9. PROBLEM ROOT CAUSE RC

The problem is the vulnerability of the customer's details to fake websites. So these websites will use the customer's details to access their bank account and loot the money.

7. BEHAVIOUR BE

The customers use phishing detection websites in order to prevent using fake websites and protect the details from those websites.

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

Identify strong TR & EM

3. TRIGGERS TR

The fear of the leakage details the customers provide triggers the customers as these details can be misused.

4. EMOTIONS: BEFORE / AFTER EM

When the customers do not use phishing detection websites they will be in the fear of the details getting leaked, scare of the money in bank account getting looted.

Once they start using phishing detection websites they will be confident in providing the details.

10. YOUR SOLUTION SL

The best solution from preventing the customers from using the fake websites is to use the phishing detection websites so they can prevent their details from getting leaked.

8. CHANNELS OF BEHAVIOUR CH

8.1 ONLINE

Customers use phishing websites in order to prevent their details that they would provide to the website from getting leaked.

8.2 OFFLINE

There will be no problem when the customer is offline as they can't use any website when they go offline.

Identify strong TR & EM

CHAPTER-4

REQUIREMENT ANALYSIS

4.1 Functional Requirements :

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail Registration by creating a newuser name and password
FR-2	User Confirmation	Confirmation viaEmail Confirmation viaOTP
FR-3	User login	Login usingthe credentials we have used during registration
FR-4	User permission	User must give permission accessto the searchengine so the intelligent system can detect phishing websites
FR-5	Using the intelligent system	User will use the intelligent system to detect phishing websites and save himself from his money being looted

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposedsolution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is very user friendly, any people with less knowledge also can easily understand that they areusing the fake website through our alert message.
NFR-2	**Security**	It is very secured as one cannot hack our detection website so one can easily trust our detection website andthey will be saved from financial and information loss.
NFR-3	**Reliability**	It has good consistency and performance as it actively detects the fake websites and protect theconfidential information and financial loss of the user.

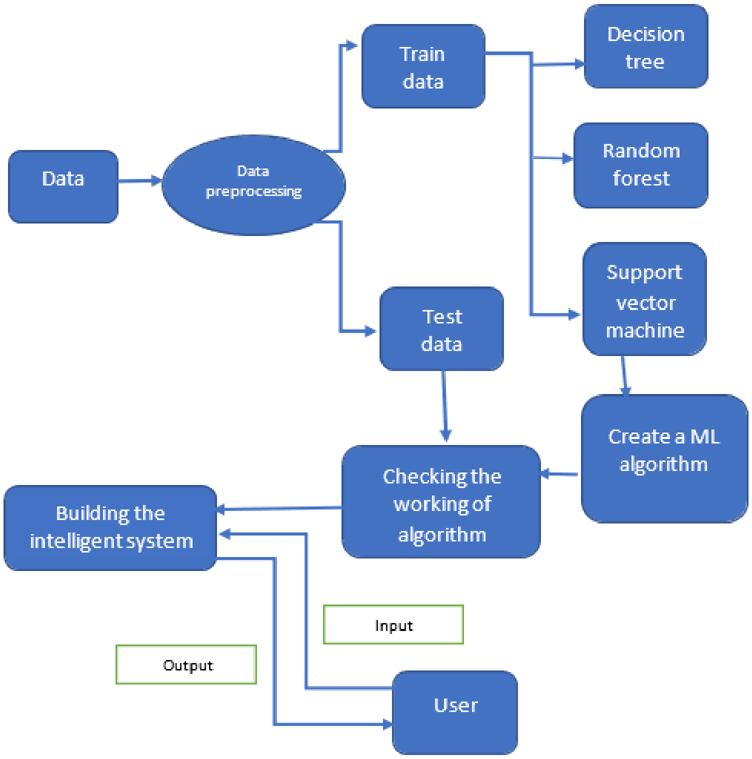
NFR-4	**Performance**	The performance of web phishing detection is highand it is very efficient as it is very easy to understand and has a high security nd scalable
NFR-5	**Availability**	This detection website is available at any system like laptop , mobile phone , desktop and user friendly
NFR-6	**Scalability**	The total execution time of our approach in phishing webpage detection is around 2-3 sec, which is quite less and acceptable environment. As input size increases the execution time increases and this makes the system difficult to handle increases the stress.

CHAPTER-5

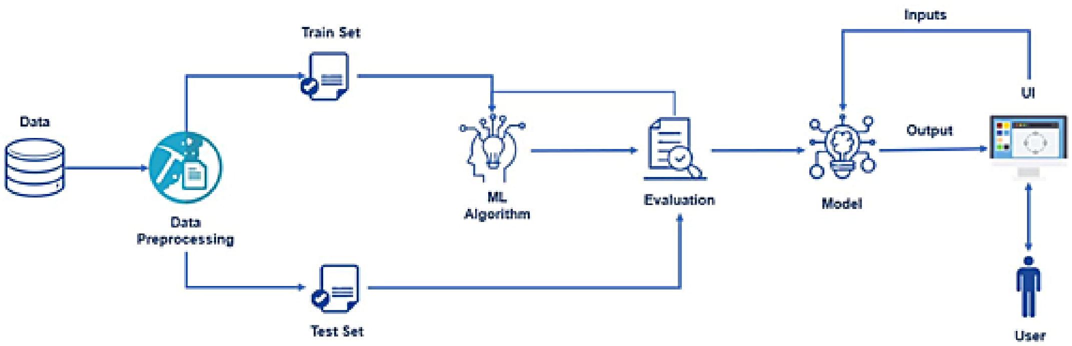
PROJECT DESIGN

5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored



5.2 Technical Architecture:

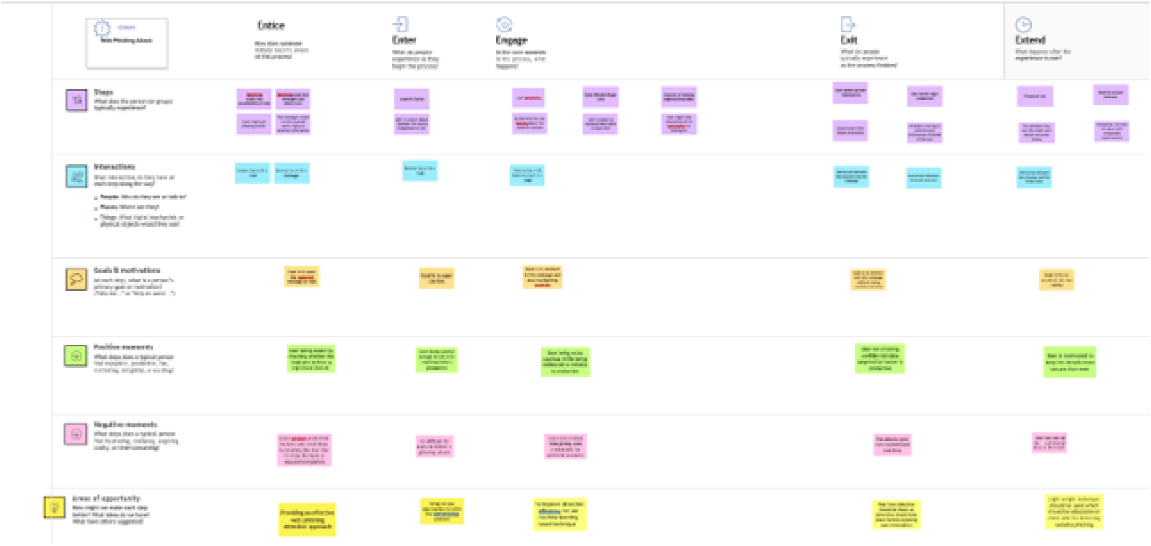


5.3 User Stories:

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

UserType	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Acceptance criteria	Priority
Customer (Mobile user)	Download the intelligent system	USN-1	As a user,I can download the intelligent systemand detect phishing websites. The system starts working immediately once youstart thecomputer.	I can download it easilyfrom internet.	High
	register	USN-2	As a user, I will register for the system using my mail and receive confirmation email once I haveregistered for the application	I can receive confirmation on email & clickconfirm	Medium
	login	USN-3	As a user, I can login to the application andentermy detailsand use the application.	I can Login and give mydetails	Medium
	Provide access	USN-4	The user should provide access to googleand theuser's search engine so that the intelligent system can detect phishing websites.	I have to provide accessto search engines	High
	use	USN-5	The user can usethe intelligent systemto detectphishing website	Finally I can use the intelli -gent system	Medium
Customer (Web user)	The functional requirements are same as mobile user.	Same as mobile user	Same as mobile user	Same as mobile user	High when compared to mobile users

Customer Journey Map:



CHAPTER-6

Milestone and activity list:

IDEATION PHASE

TITLE	DESCRIPTION	DATE
Literature Survey	Literature survey on the selected project& gathering information by referring the, technical papers,research publications etc.	8 SEPTEMBER 2022
Empathy Map for Web Phishing Detection	Prepare Empathy Map Canvasto capture the user Pains & Gains,Prepare list of problem Statements	21 SEPTEMBER 2022
Problem Statement	Prepare the problem statement document	22SEPTEMBER 2022
Brainstorming Idea Generation Prioritization	List the by organizing the brainstorming session and prioritize the top 3 ideas based onthe feasibility &importance.	24 SEPTEMBER 2022

PROJECT DESIGN PHASE-I

TITLE	DESCRIPTION	DATE
Problem SolutionFit	Prepare problem - solutionfit document.	1 OCTOBER 2022
Proposed Solution	Prepare the proposed solution document, which includesthe novelty, feasibility of idea, business model, social impact,scalability of solution, etc.	5 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	5 OCTOBER 2022

PROJECT DESIGN PHASE-II

TITLE	DESCRIPTION	DATE
Solution Requirements	Prepare the functional requirement document.	**12OCTOBER 2022**
Customer JourneyMap	Prepare the customer journey maps to understand the user interactions & experienceswith the application (entry to exit).	**14OCTOBER 2022**
DataFlow Diagrams and User Stories	Draw the data flow diagrams and submit for review.	**15 OCTOBER2022**
Technology Stack	Prepare the technology architecture diagram	**15OCTOBER 2022**

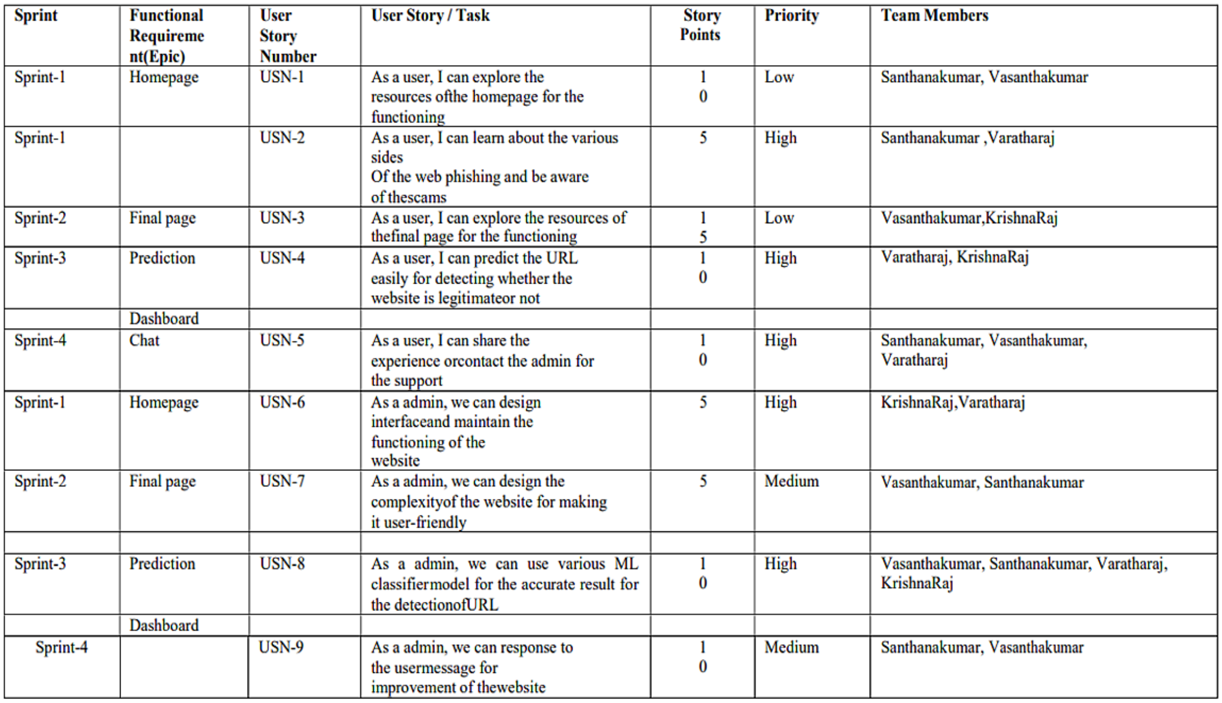
PROJECT PLANNING PHASE

TITLE	DESCRIPTION	DATE
Project Planning	Prepare the planning for this project	**27OCTOBER 2022**
Milestone and Activity List	Prepare the milestones &activity list of the project	**27OCTOBER 2022**

PROJECT DEVELOPMENT

TITLE	DESCRIPTION	DATE
Project Development Delivery of Sprint-1, 2, 3 &4	Develop & submit the developed code by testingit	**INPROGRESS**

Product backlog and sprint schedule:



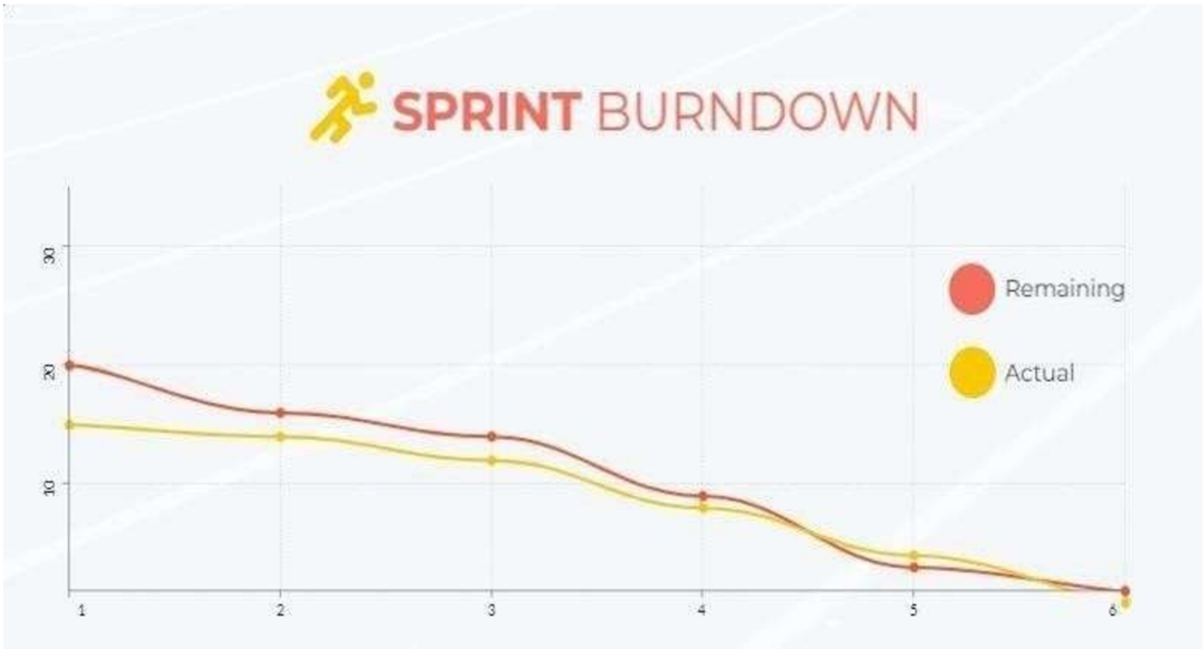
Sprint	Functional Requirement(Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	As a user, I can explore the resources ofthe homepage for the functioning	10	Low	Santhanakumar, Vasanthakumar
Sprint-1		USN-2	As a user, I can learn about the various sides Of the web phishing and be aware of thescams	5	High	Santhanakumar ,Varatharaj
Sprint-2	Final page	USN-3	As a user, I can explore the resources of thefinal page for the functioning	15	Low	Vasanthakumar,KrishnaRaj
Sprint-3	Prediction	USN-4	As a user, I can predict the URL easily for detecting whether the website is legitimateor not	10	High	Varatharaj, KrishnaRaj
	Dashboard					
Sprint-4	Chat	USN-5	As a user, I can share the experience orcontact the admin for the support	10	High	Santhanakumar, Vasanthakumar, Varatharaj
Sprint-1	Homepage	USN-6	As a admin, we can design interfaceand maintain the functioning of the website	5	High	KrishnaRaj,Varatharaj
Sprint-2	Final page	USN-7	As a admin, we can design the complexityof the website for making it user-friendly	5	Medium	Vasanthakumar, Santhanakumar
Sprint-3	Prediction	USN-8	As a admin, we can use various ML classifiermodel for the accurate result for the detectionofURL	10	High	Vasanthakumar, Santhanakumar, Varatharaj, KrishnaRaj
	Dashboard					
Sprint-4		USN-9	As a admin, we can response to the usermessage for improvement of thewebsite	10	Medium	Santhanakumar, Vasanthakumar

Project Tracker, Velocity & Burndown Chart

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

Burndown Chart:

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



CHAPTER-7

CODE

App.py

```
#importing required libraries

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

import inputScript

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

#Redirects to the page to give the user input URL.
@app.route('/')
def predict():
    return render_template('index.html',result="")

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

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

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=False)
```

inputScript.py

```
import regex
from tldextract import extract
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import requests
import favicon
import re
from googlesearch import search

#checking if URL contains any IP address. Returns -1 if contains else returns 1
def having_IPhaving_IP_Address(url):
     match=regex.search(
  '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-
4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  #IPv4
                '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-
F]{1,2})\\/)'  #IPv4 in hexadecimal
                '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)     #Ipv6
    if match:
      #print match.group()
      return -1
    else:
      #print 'No matching pattern found'
      return 1

#Checking for the URL length. Returns 1 (Legitimate) if the URL length is less than 54
characters
#Returns 0 if the length is between 54 and 75
#Else returns -1;
def URLURL_Length (url):
   length=len(url)
   if(length<=75):
       if(length<54):
          return 1
       else:
          return 0
   else:
      return -1

#Checking with the shortening URLs.
#Returns -1 if any shortening URLs used.
#Else returns 1
def Shortining_Service (url):

match=regex.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',url)
    if match:
        return -1
    else:
        return 1

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

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

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

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

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

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

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

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

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

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

# HTTPS token in part of domain of URL returns -1, otherwise returns 1
def HTTPS_token(url):
    match=re.search('https://|http://',url)
    if (match.start(0)==0):
        url=url[match.end(0):]
    match=re.search('http|https',url)
    if match:
        return -1
    else:
        return 1


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

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

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

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

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

#:% of URL of anchor<31% returns 1, % of URL of anchor ≥ 31% and ≤ 67% returns 0, otherwise returns -1
def URL_of_Anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

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

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

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

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

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

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

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

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

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

    except:
        return 0


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

        no_of_script =0
        for meta in soup.find_all(onmouseover=True):
            no_of_script = no_of_script+1
        if(no_of_script==0):
            return 1
        else:
            return -1
    except:
        return -1

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

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

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

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

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

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

#website rank < 100.000 returns 1, website rank > 100.000 returns 0, otherwise returns -
1
def web_traffic(url):
    try:
        rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" +
url).read(), "xml").find("REACH")['RANK']
    except TypeError:
        return -1
    rank= int(rank)
    if (rank<100000):
        return 1
    else:
        return 0

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

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

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

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

#:host in top 10 phishing IPs or domains returns -1, otherwise returns 1
def Statistical_report (url):
    hostname = url
    h = [(x.start(0), x.end(0)) for x in
regex.finditer('https://|http://|www.|https://www.|http://www.', hostname)]
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in regex.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:
            hostname = hostname[:h[0][0]]

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

ip_match=regex.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.
185\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.4
0|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\.8
3\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\.71|23\.25
3\.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\.10
4\.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\.127\.157|34\.196\.13\.28|10
3\.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|2
16\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|5
4\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip_address)
    except:
        return -1

    if url_match:
        return -1
    else:
        return 1

#returning scrapped data to calling function in app.py
def main(url):



    check = [[having_IPhaving_IP_Address
(url),URLURL_Length(url),Shortining_Service(url),having_At_Symbol(url),

double_slash_redirecting(url),Prefix_Suffix(url),having_Sub_Domain(url),SSLfinal_State(u
rl),

Domain_registeration_length(url),Favicon(url),port(url),HTTPS_token(url),Request_URL(u
rl),

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abnormal_URL(
url),
            Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(url),

age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Google_Index(url),
            Links_pointing_to_page(url),Statistical_report(url)]]


    return check
```

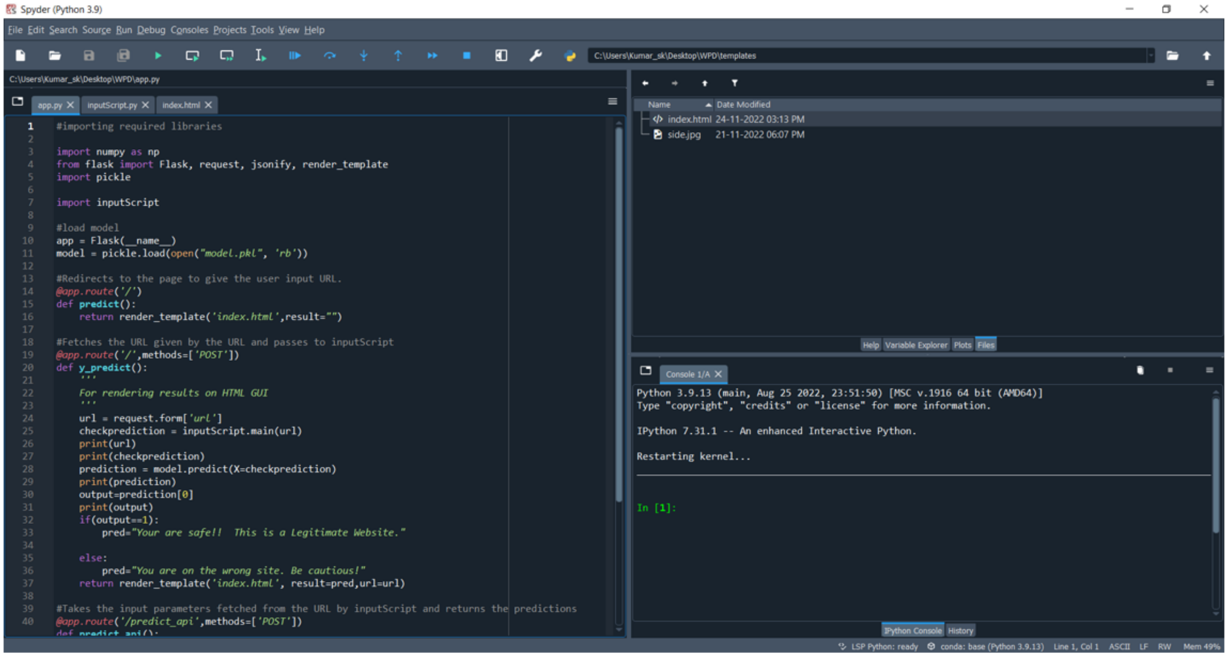
Index.html

```
<!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">

    <!-- 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/styles.css" rel="stylesheet">
    <link rel="icon" href="favicon.ico" >
    <title>Web Phising Detection</title>
</head>

<body class="bg-dark">
<div class="container mt-5">
    <div>
        <center>
        <div class="form col-md text-light" id="form1">
            <center>
            <h2>Phishing website Detection</h2>
            <br>
            <form action="/" method ="post" autocomplete="off">
                <input type="text" class="form-control w-50" name ='url' id="url"
placeholder="Enter URL" required="" />
                <br>
                <button class="btn btn-info mt-2" role="button" >Check here</button>
            </form>
        </div>
        <br>
        <div class="col-md" id="form2">
            <br>
            <h4 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h4>
            <br>
            <h3 id="prediction" class="text-warning"></h3>
            <button class="btn btn-warning" id="btn1" role="button"
onclick="window.open('{{url}}')" target="_blank">Continue to Site</button>
        </div>
        </center>
    </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 defer>
    document.querySelector("#btn1").style.display = "none";
    let result = '{{result}}';
    if(result!==undefined || result!==null){
      console.log(result)
      document.getElementById("prediction").innerHTML = result;
      document.getElementById("btn1").style.display="inline-block";
    }
  </script>

</body>
</html>
```

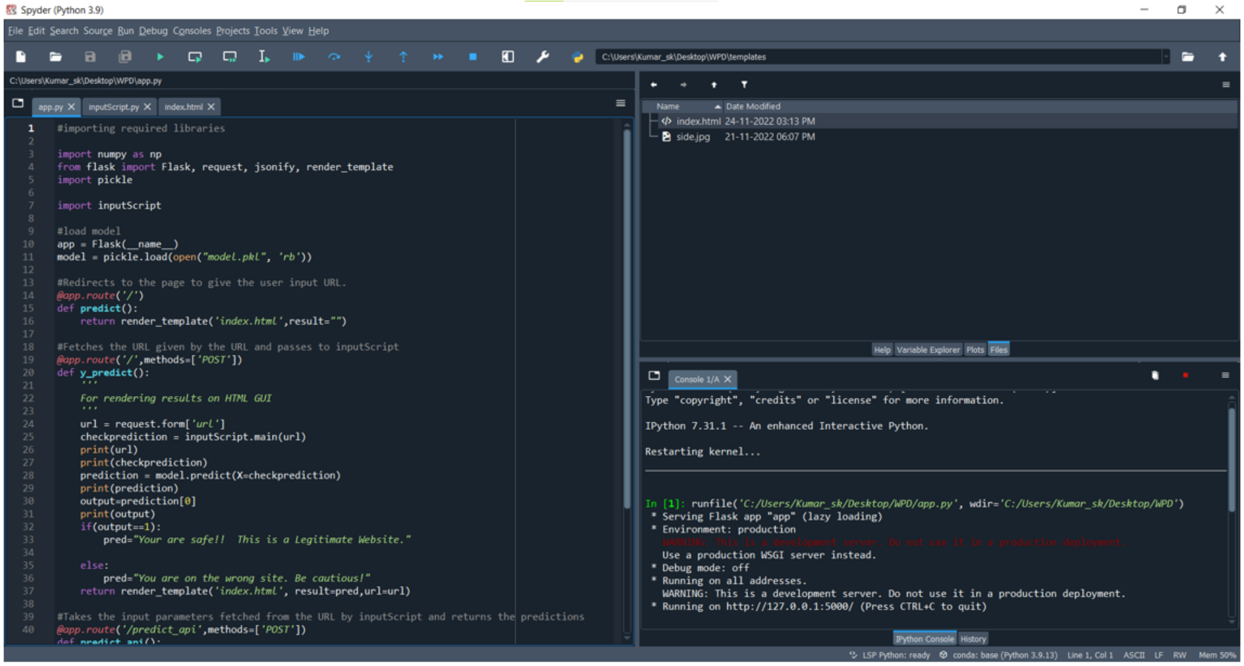
CHAPTER-8

RESULT


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Kumar_sk\Desktop\WPD\templates
C:\Users\Kumar_sk\Desktop\WPD\app.py
app.py
inputScript.py
index.html
Name
Date Modified
index.html 24-11-2022 03:13 PM
side.jpg 21-11-2022 06:07 PM
Help
Variable Explorer
Plots
Files
Console 1/A
IPython Console
History
LSP Python: ready
conda: base (Python 3.9.13)
Line 1, Col 1
ASCII
LF
RW
Mem 49%

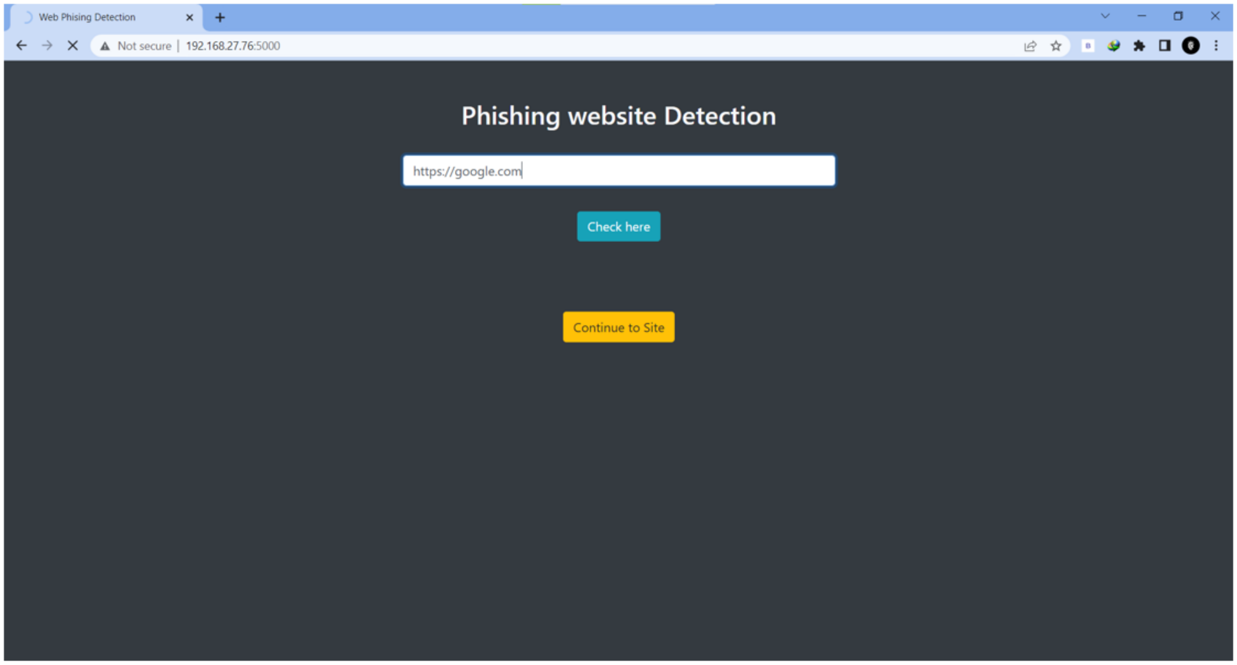
```python
#importing required libraries

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

import inputScript

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

#Redirects to the page to give the user input URL.
@app.route('/')
def predict():
    return render_template('index.html',result="")

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

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('index.html', result=pred,url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
```

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

Restarting kernel...

In [1]:
```


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Kumar_sk\Desktop\WPD\templates
C:\Users\Kumar_sk\Desktop\WPD\app.py
app.py
inputScript.py
index.html
Name
Date Modified
index.html 24-11-2022 03:13 PM
side.jpg 21-11-2022 06:07 PM
Help
Variable Explorer
Plots
Files
Console 1/A
IPython Console
History
LSP Python: ready
conda: base (Python 3.9.13)
Line 1, Col 1
ASCII
LF
RW
Mem 50%

```python
#importing required libraries

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

import inputScript

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

#Redirects to the page to give the user input URL.
@app.route('/')
def predict():
    return render_template('index.html',result="")

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

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('index.html', result=pred,url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
```

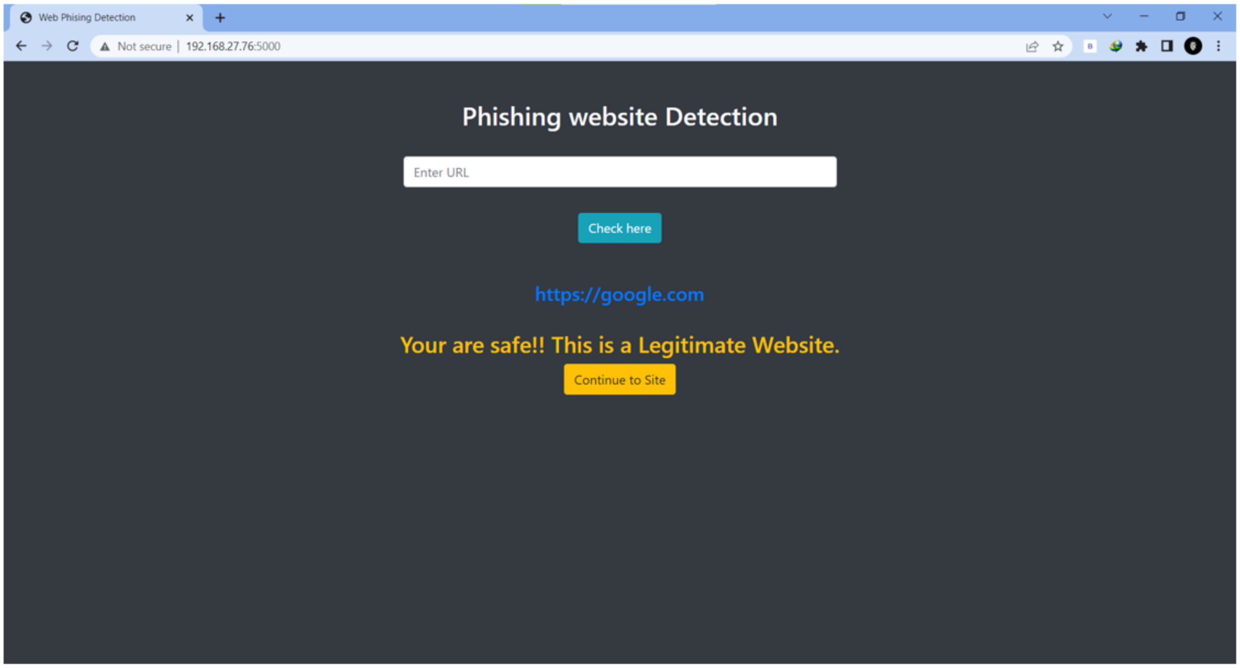
```
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

Restarting kernel...

In [1]: runfile('C:/Users/Kumar_sk/Desktop/WPD/app.py', wdir='C:/Users/Kumar_sk/Desktop/WPD')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Phishing website Detection

https://google.com

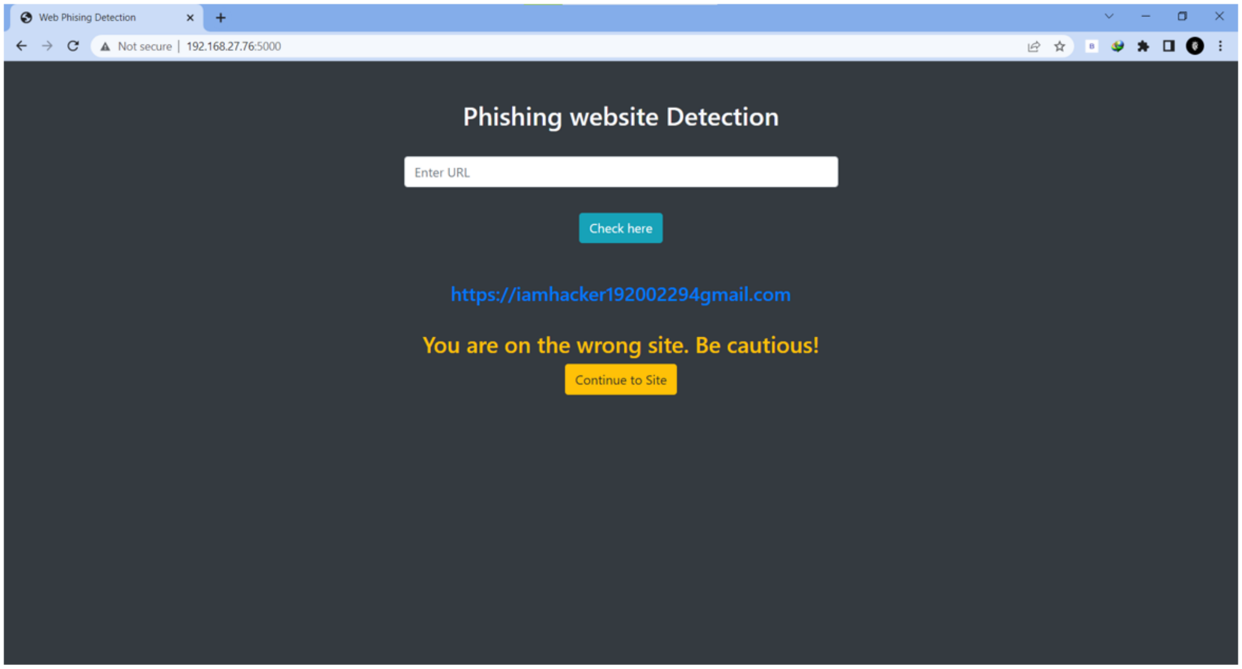
..



Phishing website Detection

https://google.com

Your are safe!! This is a Legitimate Website.



Phishing website Detection

https://iamhacker192002294gmail.com

You are on the wrong site. Be cautious!

CHAPTER-9

ADVANTAGES OF WEB PHISHING DETECTION

1. Improve on Inefficiencies of SEG and Phishing Awareness Training 2. It Takes a Load off the Security Team
3. It Offers a Solution, Not a Tool
4. Separate You from Your Competitors
5. This system can be used by many e-commerce websites in order to have good customer relationships.
6. If internet connection fails this system will work

DISADVANTAGES OF WEB PHISHING DETECTION

1. All website related data will be stored in one place.
2. It is a very time-consuming process.

CHAPTER-10

CONCLUSION

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to tackle a wide assortment of classification issues, the procedure of finding the ideal structure is very difficult, and much of the time, this structure is controlled by experimentation. Our model takes care of this issue via computerizing the way toward organizing a neural system conspire; hence, on the off chance that we construct an enemy of phishing model and for any reasons we have to refresh it, at that point our model will encourage this procedure, that is, since our model will mechanize the organizing procedure and will request scarcely any client defined parameters.

CHAPTER-11

Future Scope

There is a scope for future development of this project. We will implement this using advanced deep learning method to improve the accuracy and precision. Enhancements can be done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

CHAPTER-12

Appendix:

1. Application Building
2. Collection of Dataset
3. Data Pre-processing
4. Integration of Flask App with IBM Cloud
5. Model Building
6. Training the model on IBM
7. Ideation Phase
8. Preparation Phase
9. Project Planning

Project Link: https://github.com/IBM-EPBL/IBM-Project-5833-1658817449
Demo link :https://drive.google.com/file/d/1DbATVqNP3r_y1UKMkGb_zGlhrUyKRPdq/view?usp=share_link