Professional Readiness for Innovation, Employability, and Entrepreneurship

PROJECT REPORT

Title	**:**	Web Phishing Detection	
Team ID	**:**	PNT2022TMID18067	
Industry mentor	**:**	Sandesh P	
Faculty mentor	**:**	Ananthi. G	
Team Lead	**:**	Saravanakumar. S	(9517201904137)
Members	**:**	Akash. M	(9517201904009)
		Purushothaman. D. S	(9517201904114)
		Rajesh. K	(9517201904119)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

This system “Web Phising detection” aims to build a machine learning model to predict phising websites by extracting feautures from the URL of the websites. Some of the notable feautures like having IP address, having at symbol, double slash redirecting, Url length etc. Model is build upon by analysing the performance of the various machine learning algorithms such as Decision Tree , LogisticRegression, SVM , Random Forest based on their accuracy metric and by simultaneously extracting features.

This system was implemented as a web application where the user enters the URl of the website to predict whether it is a phising website or not.

1.2 Purpose

Nowadays Phishing becomes a main area of concern for security researchers because it is not difficult to create the fake website which looks so close to legitimate website. Experts can identify fake websites but not all the users can identify the fake website and such users become the victim of phishing attack. Main aim of the attacker is to steal banks account credentials by phising e banking websites . Phishing attacks are becoming successful because lack of user awareness. Since phishing attack exploits the weaknesses found in users, it is very difficult to mitigate them. Hence there is need for a detection system which detects these phising websites.

2.LITERATURE SURVEY

2.1 Existing problem

The commonly used detection methods for phishing webpages, based on blacklist detection and Content-based detection methods first need to obtain web content, and then judge the legitimacy of the web page to be tested based on the similarity of web content or machine learning technology. This method needs to obtain web content, which increases the risk of the client. In addition, it requires a lot of manual feature engineering. Many of the features need to be confirmed by relevant experts. Its performance depends heavily on the quality of the manually extracted features. The detection model is easily bypassed by phishing attackers due to relatively fixed characteristics.

2.2 References

Title	Author Name	Published year	Description
Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning.	Ubing, Alyssa Anne et al.	2019	● This proposed work is capable of improving the accuracy of phishing website detection, Since a feature selection algorithm is used and integrated with an ensemble learning methodology which is a machine learning technique that combines several base models in order to produce one optimal predictive model. ● The experimental results prove that the accuracy rate of our proposed model can yield up to 95%, which is higher than the current technologies for phishing website detection
An Assessment of Features Related to Phishing Websites using an Automated Technique	Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi	2012	● This research aims to develop a group of features that have been shown to be effective in predicting phishing websites and to extract those features according to new scientific precise rules. ● Every feature will be associated with a weight corresponding to the ratio of that feature in the data collection.

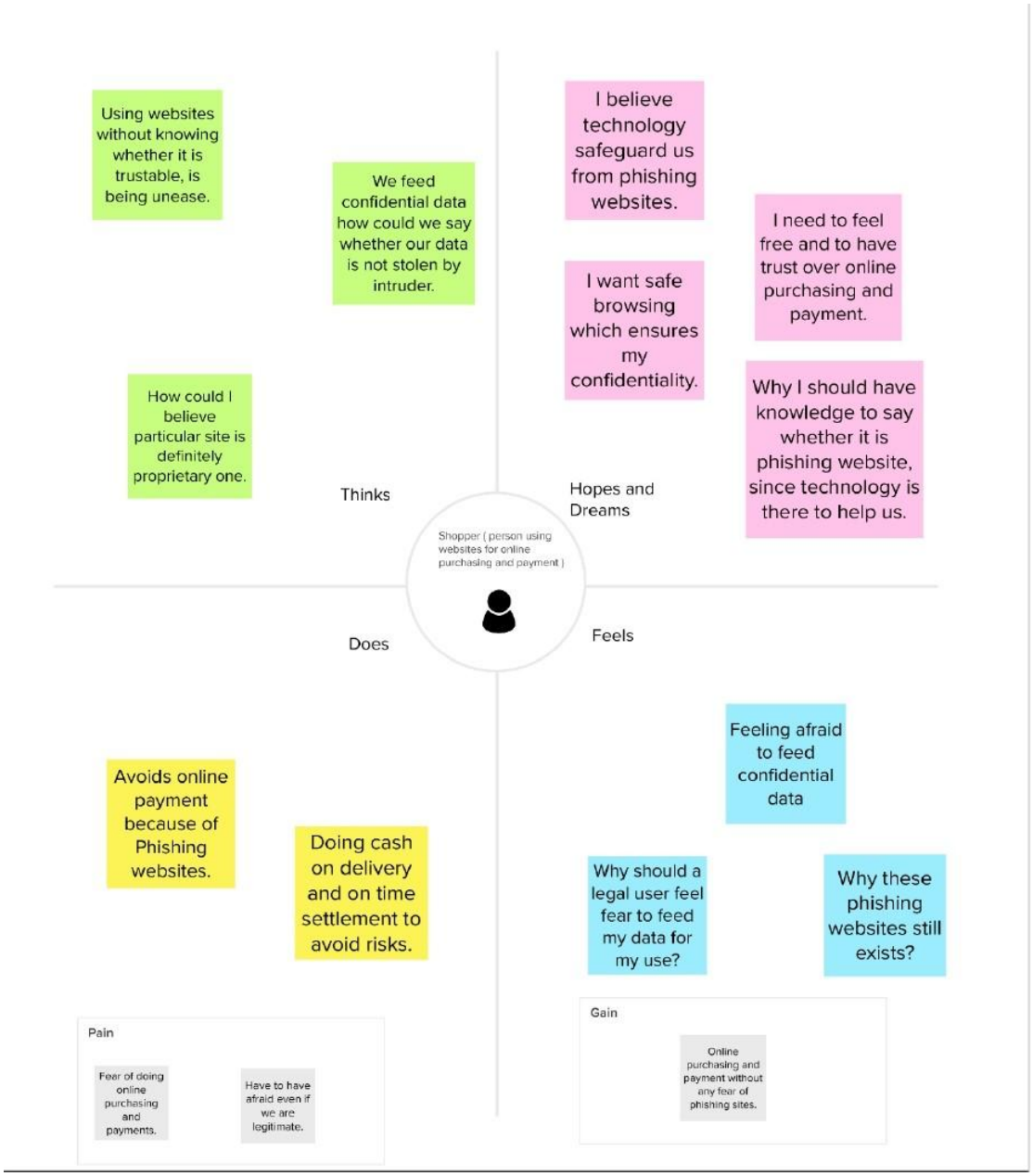
A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework,	S. Patil and S. Dhage	2019	• In this proposed work 5 major anti-phishing solutions namely, Heuristic Based Approach, Content Based Approach, Blacklist Based Approach, Machine Learning Approach, Hybrid Approach is discussed So these approaches acts as to outline a framework that can give assurance from phishing attacks.
A phishing vulnerability analysis of web based systems	W. D. Yu, S. Nargundkar, and N. Tiruthani	2008	• In this article they have addressed some of the phishing vulnerabilities of web based systems namely Browser Vulnerabilities , Misleading emails, Exploitable Security holes, unverified source address etc. • It also discusses some of the attempts made to prevent phishing such as Spam filters,browser toolbar extensions to alert the user about phishing websites, augmenting password logins, maintaining databases with phishing websites and monitoring and taking down phishing websites.

2.3 Problem Statement Definition

The web applications are the primary target for the unethical hackers. One of the methods helps such attacker is web phishing. Commercial websites such as online purchasing platforms, online banking sites are handling with lot of sensitive informations. Hence there is a need for an intelligent system which detects the phishing websites especially for the e-banking phishing sites.

3.IDEATION & PROPOSED SOLUTION

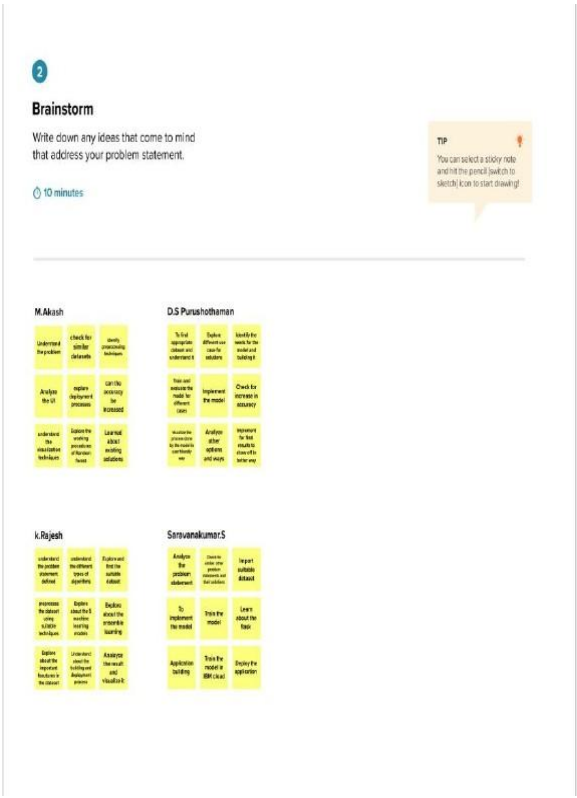
3.1 Empathy Map Canvas

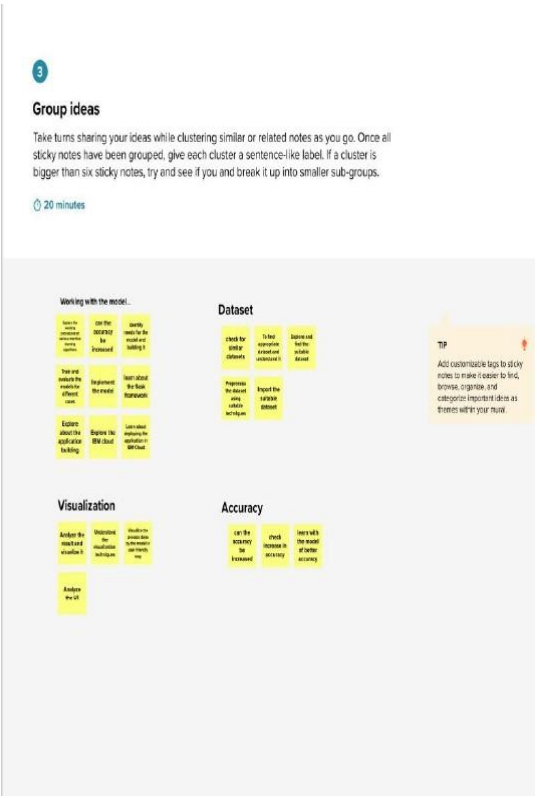


3.2 Ideation & Brainstorming

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

Step-2: Brainstorm, Idea Listing and Grouping





Step-3: Idea Prioritization

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The web applications are the primary target for the unethical hackers. One of the methods helps such attacker is web phishing. Commercial websites such as online purchasing platforms, online banking sites are handling with lot of sensitive informations. Hence there is a need for an intelligent system which detects the phishing websites especially for the e-banking phishing sites.
2.	Idea / Solution description	Machine Learning model takes URL as input. In that some of the notable features such as urllength,having ip address,port etc... Using these features the model is trained and it detects whether the website is phished website or not.
3.	Novelty / Uniqueness	Existing solutions for Web Phishing detection mainly focused on some of the anti-phishing solutions namely, Heuristic Based Approach, Content Based Approach, Blacklist Based Approach. But our proposed solution is based on a machine learning approach which is more efficient and accurate than existing approaches.
4.	Social Impact / Customer Satisfaction	Many people when they need to purchase something or to do an online banking they rely on websites and they don't know whether the website is a reliable one or a phishing website. So having an phishing website detector will provide a safe and satisfactory experience for the users to share their datas.
5.	Business Model (Revenue Model)	If there is phising detector the datas given by the users will not be theft and used for an illegal purpose which reduces the workload of cyber crime police officers. Also the phising detector can be added as an extension to every browsers and can generate revenue through charging for the extension.
6.	Scalability of the Solution	Since the phising detctor can be added as an extension or individual website it can be used in any kind of operating system and performs equally well with one or a thousand users and stands ups and downs of the traffic. Also easily movable by simply adding as an extension in desired browser.

3.4 Problem Solution fit

<table>
<tr>
<td><u>1. Customer Segments</u>
+ Website users</td>
<td><u>6. Customer Limitation</u>
Since it is time consuming process to undergo the system of phising detection customers who in need of quick response from the website may hesitate use this feature.</td>
<td><u>5. Available Solution</u>
Existing solutions mainly focuses on Content Based Approach and predefined rules but there may be some hidden rules and may fail to detect some of the phishing sites .</td>
</tr>
<tr>
<td><u>2. Problems</u>
Since every website needs to undergo the system of phising detection , conventional browsing is interrupted in terms of response time.</td>
<td><u>9. Problem root cause</u>
No awarness about the phising websites among the users. Also there is no proper platform that assures the webphising detection.</td>
<td>7. <u>Behavior</u>
Customers are supposed to enter the URL of the website in the web application to detect whether the website is phished or ot.</td>
</tr>
<tr>
<td><u>3. Triggers to Act</u>
When customers wants to give their datas safely in the website to avoid the theft of sensitive information of them.</td>
<td rowspan="2"><u>10. Your Solution</u>
Machine Learning model takes URL as input. In that some of the notable features such as urllength,havingip address,port etc... Using these features the model is trained and it detects whether the website is phished website or not.</td>
<td><u>8. Channels of Behavior</u>
1. Online: URL of the website need to be entered in the web application.</td>
</tr>
<tr>
<td><u>4. Emotions</u>
When customers gives their datas in the website they may feel insecure that their datas may be stealed. After using our product they will feel confident that datas will not be misused.</td>
<td>2. Offline: No actions need to be taken by the customers in the offline</td>
</tr>
</table>

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	**Accepts User's inputs**	The system must accept the user input which is entered via smart phone / desktop .
FR-2	**Process User's inputs**	The system must give the user inputs as the test inputs to the model built.
FR-3	**Analysis**	The system must allow the model to perform analysis on the inputs.
FR-4	**Prediction**	The system must allow the model to predict the performance based on the training data.
FR-5	**Output**	The system must display the prediction value to the user.

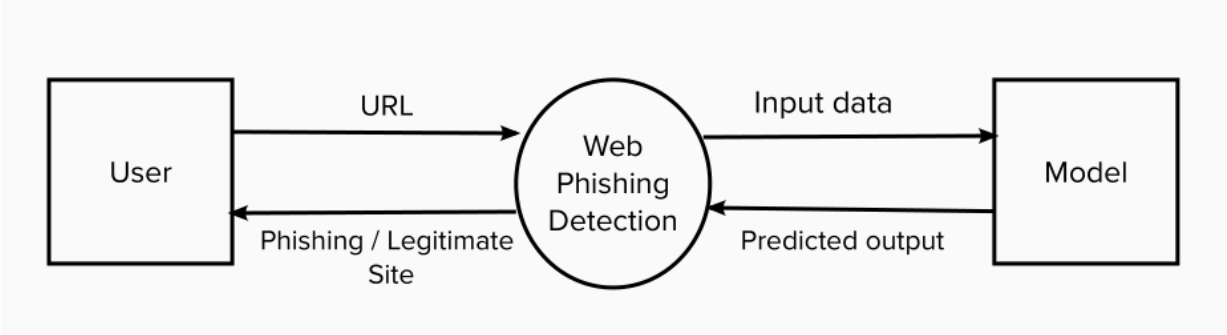
4.2 Non-functional Requirements:

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

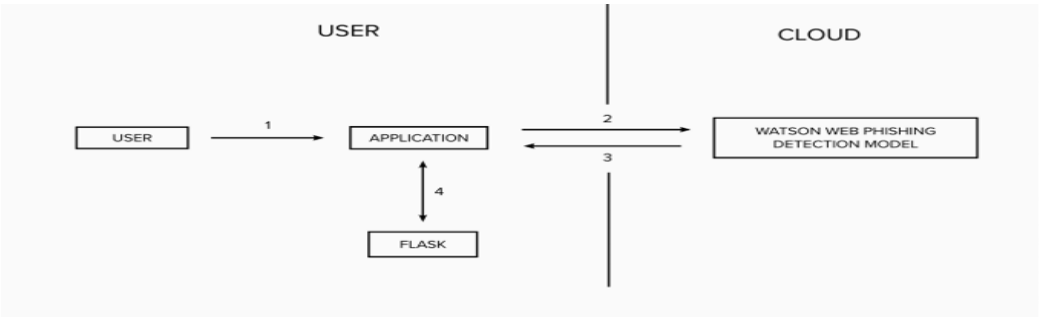
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user interface will be user-friendly for users to use.
NFR-2	**Security**	Users can access the application 99.9% of the time without failure
NFR-3	**Reliability**	The application provides better performance by quick loading of the application.
NFR-4	**Performance**	It describes how likely the system is accessible to a User at a given point of time. The application is available whenever the user needs it.
NFR-5	**Availability**	Able to run on any Operating Systems.
NFR-6	**Scalability**	Able to update the model to predict the more accurate phishing sites precisely.

5.PROJECT DESIGN

5.1 Data Flow Diagrams:



5.2 Solution & Technical Architecture:



1. Initially user starts the application which is developed using flask framework and enters the URL which is going to predict whether it is phishing site or not.
2. The URL is given as the input to the model which is trained on IBM Watson cloud platform for prediction.
3. Then the model predicts the output based on the input URL.
4. Finally the predicted output is displayed in the User Interface which integrated with Flask framework.

5.3 User Stories:

Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Accepts/Enable user's input	USN-1	As a user, I can enter into the application.	I can access the application.	High	Sprint-1
	USN-2	As a user, I can give the inputs to the application.	I can view the URL in the input box and the system must accepts it.	High	Sprint-1
Process user's inputs/ Prediction	USN-3	As a user, I can predict the website whether it is phishing or not based on the URL parameters.	It will help the user to don't need to enter into the phishing sites.	High	Sprint-4

6.Project Planning & Scheduling

6.1 Sprint Planning and Estimation

Title	Description	Date
Literature Survey	Gathering Information by referring the technical papers, research publications etc.	3 September 2022
Prepare Empathy Map	To capture user pain and gains Prepare List of Problem Statement	10 September 2022
Ideation	Prioritize a top 3 ideas based on feasibility and Importance	17 September 2022
Proposed Solution	Solution include novelty, feasibility, business model, social impact and scalability of solution	24 September 2022
Problem Solution Fit	Solution fit document	1 October 2022
Solution Architecture	Solution Architecture	1 October 2022
Customer Journey Map	To Understand User Interactions and experiences with application	8 October 2022
Functional Requirements	Prepare functional Requirement	12 October 2022
Data flow Diagrams and User Stories	Data flow diagram	12 October 2022
Technology Architecture	Technology Architecture diagram	12 October 2022
Milestone & sprint delivery plan	Activity what we done &further plans	22 October 2022
Project Development- Delivery of sprint 1,2,3 &4	Develop and submit the developed code by testing it	24 October 2022 – 19 November 2022

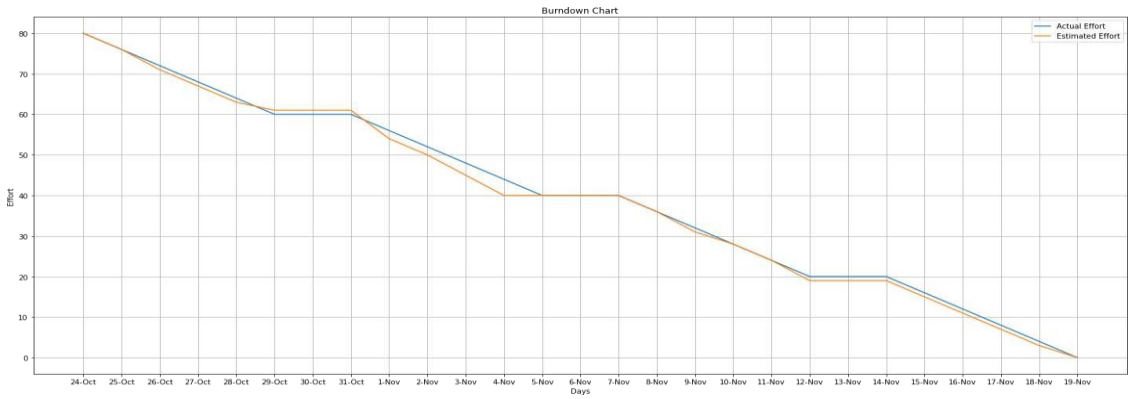
6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Interface	USN-1	As a user, I need UI to do what I need to do.	20	Low	Purushothaman D S
Sprint-2	Url Entry	USN-2	As a user, I Should be able to feed url as input.	20	Medium	Rajesh K
Sprint-3	Feature Extraction	USN-3	As a user, I need the application to exactly get necessary details from the website for prediction.	20	Medium	Akash M
Sprint-4	Phishing Site Prediction	USN-4	As a user, I expect the application to differentiate legitimate site and phishing website.	20	Medium	Saravana Kumar S

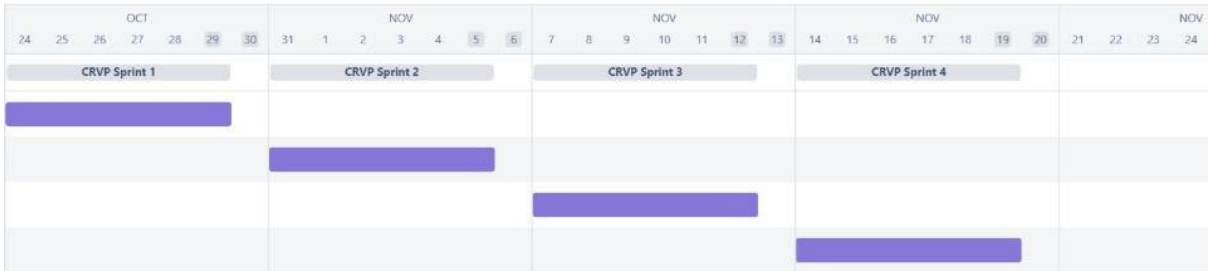
6.3 Project Tracker

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on planned end date)	Sprint Release Date (Actual)
Sprint1	20	6 days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint2	20	6 days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint3	20	6 days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint4	20	6 days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.4 Burndown Chart

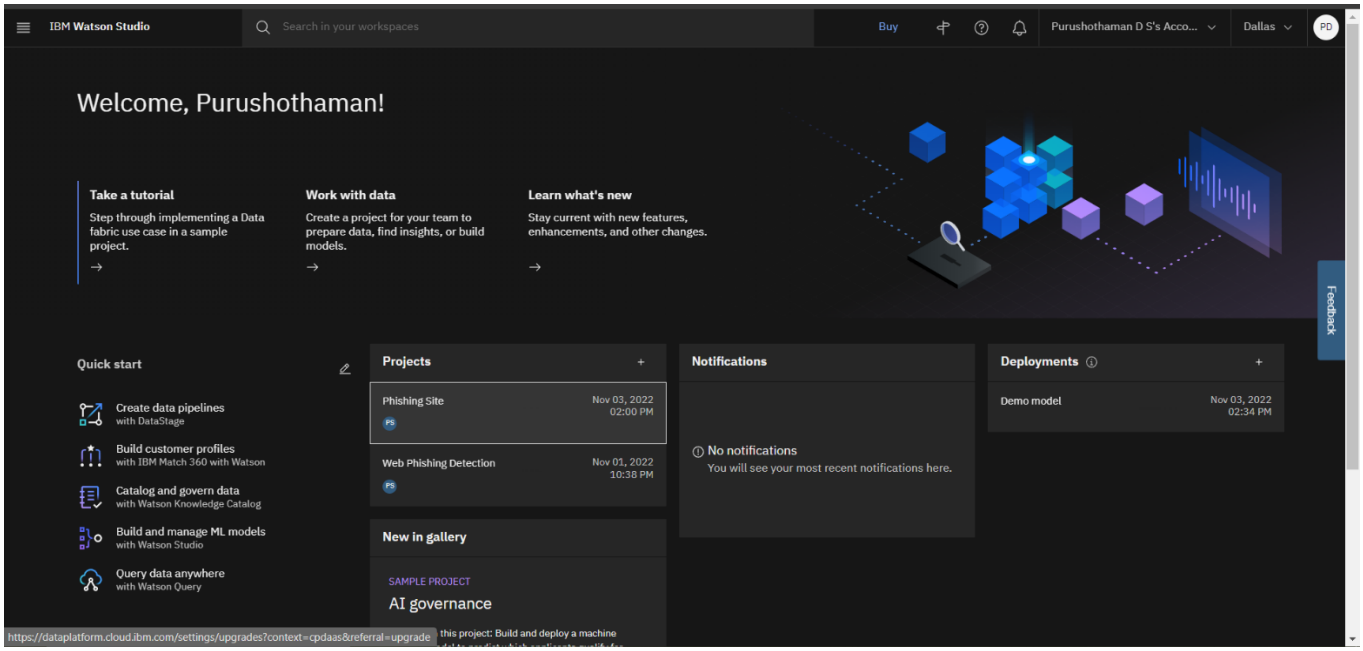


6.5 Reports from JIRA

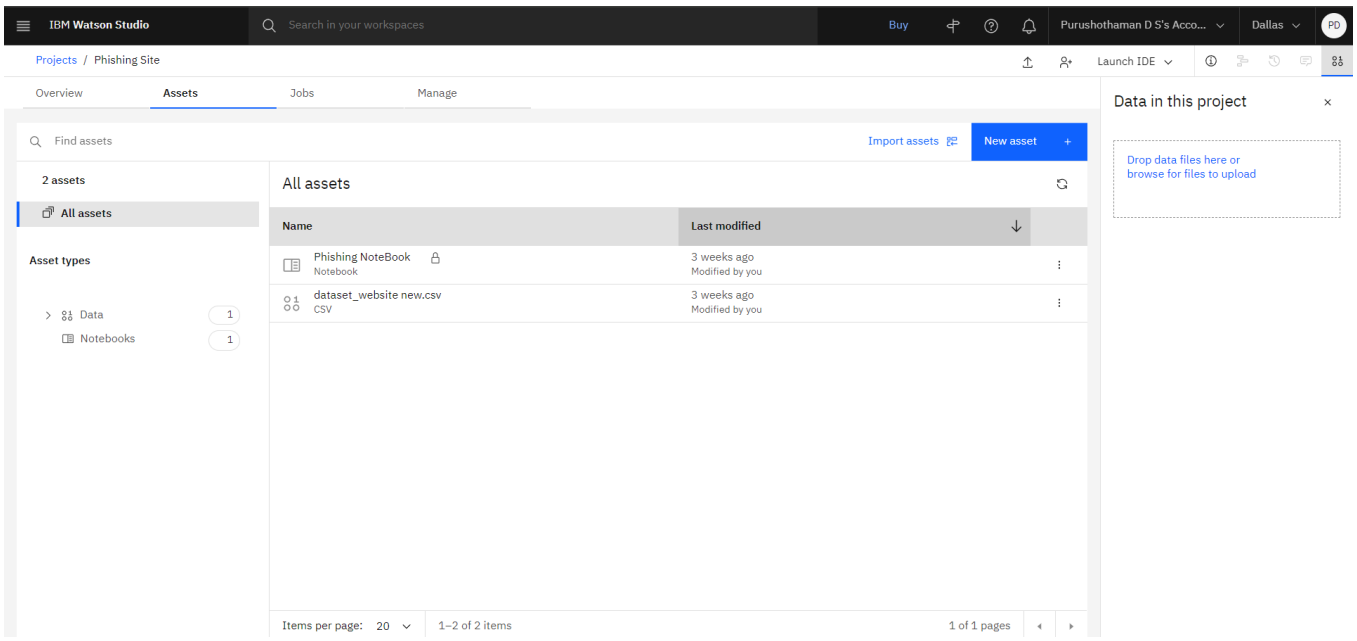


7.CODING & SOLUTIONING

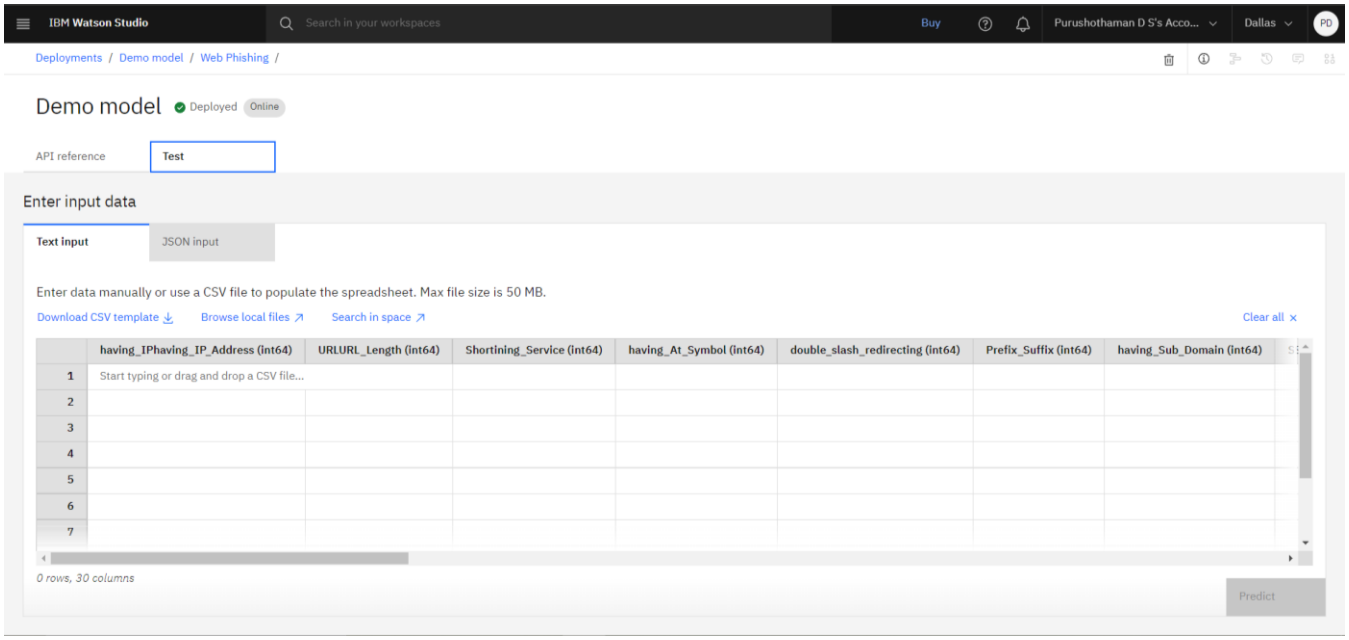
7.1 Creating IBM Watson Project :



7.2 Adding Notebook and Data in IBM Watson :



7.3 Creating Deployment Model and Testing in IBM Watson :



8.Testing

8.1 Test Case Scenarios

Testcase Scenario Id	Testcase Scanario
WPD_TC_1	Verify, whether the user can able to find what they need to do to identify the correctness of website.
WPD_TC_2	Verify, whether the submitted data is properly feeded to the feature extraction engine
WPD_TC_3	Verify, the correctness of feature extraction engine
WPD_TC_4	Verify, whether the results of feature extraction engine is in correct order as needed for machine learning model
WPD_TC_5	Checking, whether the model correctly identifies the phishing site
WPD_TC_6	Checking, whether the model correctly identifies the legitimate site
WPD_TC_7	Whether the results produced by the model is shown clearly
WPD_TC_8	Whether the final page allows the user to get to the home page.

8.2 User Acceptance Testing

Testcase Id	Feature Type	Component	Test Scanario	Pre-Requisite	Steps to Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation	BUG ID	Executed By
WPD_TC_1	Functional	Home Page	Verify, whether the user can able to find		1.get to the home page of system 2.check for interactiveness of web	No test case needed	The good user interactiveness	Having as expected	success	NIL	N	-	Rajesh k, Akash M
WPD_TC_2	Functional	Home Page Submission	Verify, whether the submitted data is properly	The url of any websites	1.enter the url in the input field. 2.press submit button	www.google.com	The url should be submitted to feature extraction engine	working as expected	success	NIL	N	-	Akash M
WPD_TC_3	Functional	Feature Extraction	Verify, the correctness of feature	The url of any websites	1.feed the url to the feature extraction engine 2.check against the	www.google.com	The correct values of extracted features	working as expected	success	NIL	N	-	Saravana Kumar S , Akash M ,
WPD_TC_4	Functional	Feature Extraction	Verify, whether the results of feature	The url of any websites and input	1.feed the url to the feature extraction engine 2.check against the	www.google.com	The correct order of extracted feature values	working as expected	success	NIL	N	-	Purushothaman D S
WPD_TC_5	Functional	Machine Learning Model	Checking, whether the model correctly	The phishing site url	1.feed the url to the model 2.verify the result	192.16.20.78:8090 /search_engine	The model is expected to correctly identify the phishing site	working as expected	success	NIL	N	-	Saravana Kumar S
WPD_TC_6	Functional	Machine Learning Model	Checking, whether the model correctly	The legitimate site url	1.feed the url to the model 2.verify the result	www.google.com	The model is expected to correctly identify the legitimate site	working as expected	success	NIL	N	-	Saravana Kumar S
WPD_TC_7	Functional	Final Page	Whether the results produced by the model is		1.feed url into input field 2.press submit 3.check the message shown in final page matches the	No testcase needed	The page is expected to show clear message received from model	Having as expected	success	NIL	N	-	Rajesh k
WPD_TC_8	Functional	Final Page	Whether the final page allows the user		1.get to final page 2.check for home button/back button	No testcase needed	The page is expected to have back/home page button	Having as expected	success	NIL	N	-	Purushothaman D S

8.3 UAT Report

8.3.1 **Defect Analysis**

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

Resolution	**Severity 1**	**Severity 2**	**Severity 3**	**Severity 4**	**Subtotal**
By Design	5	3	1	2	**11**
Duplicate	1	0	3	1	**5**
External	3	2	1	1	**7**
Fixed	10	1	3	15	**29**
Not Reproduced	0	0	0	1	**1**
Skipped	0	1	2	1	**4**
Will not Fix	0	5	2	1	**8**
Totals	**19**	**12**	**12**	**22**	**65**

9.Results

9.1 Performance metrics:

The Intelligent system for web phishing detection was implemented as machine learning model and evaluated using accuracy evaluation metrics. The reason for choosing accuracy metrics is that, we need the intelligent system to be accurate in both the decisions (Phishing and legitimate site) since misprediction of any of the situations may lead to high level threat. The accuracy of our intelligent system is 96.74 percentage.

10. ADVANTAGES & DISADVANTAGES

Advantages :

- Protectivity of personal Data, because it helps the user to identify the phished website, so that the user aware about the website. Websites like online shopping will ask for the personal details the user can detect whether the website is trust worthy or not and can protect their data.
- No need of technical expertise, all users can use this website to detect the phished website, the user need to give the url of the website as a input. There is no need of any technical knowledge.
- Using this Large organization can escape from traps like kind of scams.

Disadvantages :

- The feature extraction techniques used here is highly dependent on external packages.
- The user needs to give the website URL as input, it doesn't take automatically.
- The build model accuracy is 96.76%, but there may be better model

11. CONCLUSION

The web applications are the primary target for the unethical hackers. One of the methods helps such attacker is web phishing. Commercial websites such as online purchasing platforms, online banking sites are handling with lot of sensitive information. Hence there is a need for an intelligent system which detects the phishing websites especially for the e-banking phishing sites. Here the features are extracted from the URL, using that a phishing website detection model has been build which helps the user to detect the phishing website and protect their confidential data.

12. FUTURE SCOPE

As per the statistics until July 2022, 5.03 billion people using internet. Nowadays even the simple site is asking to create account for using the site, in which they collect the users confidential data. Using a phished website not only they collect the confidential data, also it can inject a malicious software or virus to the system. Using that hacker can get access to the system without the knowledge of the user. In the coming years, there may be many types of attacks can be initiate from the phished website, so using web phishing detection the user can prevent their system from attacks and protect their confidential data from the unauthorized individuals.

13. APPENDIX

Flask Python File :

```
import requests
API_KEY = "p0TDrnvV_e4AAWxKBpT5JKScRH_exx1tDXyfMl0dIyEg"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

import new_input
import numpy as np
from flask import Flask , request , jsonify , render_template
import pickle
from feature import FeatureExtraction
app = Flask(__name__,template_folder='template')
model = pickle.load(open('web_phishing_detector.pkl','rb'))
@app.route ('/predict')
def predict():
    return render_template ('final.html')
#Fetches the URL given by the URL and passes to inputScript
@app.route ( '/y_predict' , methods = ['POST'])
def y_predict ( ) :
    print("In Y_Predict")
    url = request.form [ 'URL' ]
    obj = FeatureExtraction(url)
    x = obj.getFeaturesList()
    print("Test :",x)

    payload_scoring = {"input_data": [{"fields": [['having_IPhaving_IP_Address',
'URLURL_Length', 'Shortining_Service',
       'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix',
       'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length',
       'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor',
       'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
       'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe',
       'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank',
       'Google_Index', 'Links_pointing_to_page', 'Statistical_report']], "values": [x]}]}

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/c1b4c9ee-fecf-4bff-970c-
cfc379ffa2e4/predictions?version=2022-11-03', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    pred=response_scoring.json()
    output = pred['predictions'][0]['values'][0][0]
    print(output)
    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 ( 'final.html' , prediction_text = 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 ( ) :
    data = request.get_json ( force = True )
    prediction = model.y_predict ( [ np.array ( list ( data.values ( ) ) ) ] )
    output = prediction [ 0 ]
    return jsonify ( output )
if __name__=='__main__':
     app.run ("127.0.0.1",5000)
```

Input Script Python File :

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

class FeatureExtraction:
    features = []
    print("Enterrrrr")
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""
        self.check = ""
        print("Enterrrrr")
        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(self.response.text, 'html.parser')
```

```
except:
    pass

try:
    self.urlparse = urlparse(url)
    self.domain = self.urlparse.netloc
    print("1")
except:
    pass

try:
    print("2")
    self.whois_response = whois.whois(self.domain)
    print("3")
except:
    print("here")
    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:
            print("11")
            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|snipu
rl\.com|'
                  'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fi
c\.kr|loopt\.us|'
                  'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|
lnkd\.in|'
                  'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\
.im|'
                  'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb
|yourls\.org|'
                  'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.
com|tweez\.me|v\.gd|tr\.im|link\.zip\.net', self.url)
        if match:
            return -1
        return 1

    # 4.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 domain in
head.link['href']:
                        return 1
            self.check = "rr"
            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 (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=
" + 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]+)",
rank_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|myjin
o\.ru|96\.lt|ow\.ly', 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|7
8\.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\.14
4\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166
\.231|216\.58\.192\.225|'
                             '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.
210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215
\.140|69\.172\.201\.153|'
                             '216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.24
3\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.
127\.102|195\.16\.127\.157|'
                             '34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|
192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.7
2|87\.98\.255\.18|209\.99\.17\.27|'
                             '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.15
8|54\.86\.225\.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
```

```
url = "https://www.google.com/"
obj = FeatureExtraction(url)
print(obj.features)
print(obj.url)
print(obj.domain)
print(obj.whois_response)
print(obj.urlparse)
print(obj.response)
print(obj.check)
```

HTML File :

```
<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">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6J
Xm" crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG
5KkN" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q
" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl
" crossorigin="anonymous"></script>
  <style>
    body{
      background-color: rgb(217,231,242);
    }
    .header{
      margin-top: 3%;
      margin-left: 2%;
      align-items: center;
    }
  </style>
  <title>Document</title>
</head>
<body>
    <div class="header">
```

```
    <img src="C:\Users\ACER\Downloads\phishing.png" alt="" width="4%"
style="display:inline-block;">
    <h3 style="display: inline-block; margin-bottom: 0;">Web Phishing
Detection</h3>
  </div>
  <div style="display: flex;">
   <div style="flex: 2;">
    <form action='http://127.0.0.1:5000/y_predict' method="post" style="margin-
top: 15%; margin-left: 35%;">
     <div class="form-group" style="width: 50%;">
      <label for="link1">Website Link</label>
      <input type="text" name="URL" class="form-control" id="link1"
placeholder="Enter Website Link">
     </div>
     <button type="submit" class="btn btn-primary">Submit</button>
    </form>
    <p>{{prediction_text}}</p>
    <p>{{url}}</p>
   </div>
   <div style="flex: 2;"><img
src="C:\Users\ACER\Downloads\8135200_3831994.jpg" alt=""
width="100%"></div>
  </div>
</body>
</html>
```

GitHub Link :
https://github.com/IBM-EPBL/IBM-Project-2443-1658471772
Demo Video Link :
https://drive.google.com/file/d/18EPzisfFViAcJlbebmx2nwp5c5NONnou/view?usp=share_link