

WEB PHISHING DETECTION

IBM-Project-43325-1660715910

**NALAIYA THIRAN PROJECT BASED ON LEARNING
PROFESSIONAL READINESS FOR
INNOVATION, EMPLOYABILITY AND
ENTREPRENEURSHIP**

Project Report

TEAM ID: PNT2022TMID47793

TEAM MEMBERS:

1. KIRUTHIKA PARAMESHWARI . M 911719104026
2. SHANMUGAPRIYA . TR - 911719104064
3. SHRREENIDHI . S - 911719104066
4. UMABHARATHI . L - 911719104077

**BACHELOR OF ENGINEERING IN
COMPUTER
SCIENCE AND ENGINEERING**

**MOUNT ZION COLLEGE OF ENGINEERING
AND TECHNOLOGY**

ANNA UNIVERSITY , CHENNAI - 600 025

INDEX

1. INTRODUCTION

1. Project Overview
2. Purpose

2. LITERATURE SURVEY

1. Existing problem
2. References
3. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Proposed Solution
4. Problem Solution fit

4. REQUIREMENT ANALYSIS

1. Functional requirement
2. Non-Functional requirements

5. PROJECT DESIGN

1. Data Flow Diagrams
2. Solution & Technical Architecture
3. User Stories

6. PROJECT PLANNING & SCHEDULING

1. Sprint Planning & Estimation
2. Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

1. Feature 1
2. Feature 2

8. TESTING

1. Test Cases
2. User Acceptance Testing

9. RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

This project mainly focuses on applying a machine-learning algorithm to detect phishing websites. In order to detect and predict 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 dataset's criteria to classify their legitimacy. The phishing website can be detected based on some important characteristics, like the URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user enters a website, our system will use a data mining algorithm to detect whether the website is a phishing website or not.

1.2 Purpose

There are a number of users who purchase products online and make payments through e-banking. Some e-banking websites ask users to provide sensitive data such as username, password, and credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web services are one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet. There are millions of incidents happening around the world in an hour. People suffer immeasurable losses due to these attacks. Therefore, protecting users from such attacks is the sole purpose of our project.

The simplest method of obtaining sensitive information from unwitting users is through phishing attacks. The goal of phishers is to obtain vital data, such as username, password, and bank account information. In this research, many properties of legal and phishing URLs are extracted and analyzed in order to detect phishing URLs. The algorithms used to identify phishing websites include decision trees, random forests, and support vector machines. By evaluating each algorithm's accuracy rate, false positive rate, and false negative rate, the study aims to identify phishing URLs as well as identify the best machine learning method.

2. LITERATURE SURVEY

2.1 Existing problem

Due to how simple it is to create a fake website that closely resembles a legitimate website, phishing has recently become a top concern for security researchers. Experts can spot fake websites, but not all users can, and those users end up falling for phishing scams. The attacker's primary goal is to steal bank account credentials. Businesses in the US lose \$2 billion annually as a result of their customers falling for phishing scams. The annual global impact of phishing was estimated to be as high as \$5 billion in the third Microsoft Computing Safer Index Report, which was published in February 2014. Because users are unaware of phishing attacks, they are becoming more successful.

Since phishing attacks take advantage of user vulnerabilities, it is highly challenging to counteract them, but it is crucial to improve phishing detection methods. The common technique, commonly referred to as the "blacklist" method, for detecting phishing websites involves

adding Internet Protocol (IP) blacklisted URLs to the antivirus database. Attackers utilize clever methods to deceive people by changing the URL to seem authentic through obfuscation and many other straightforward tactics, such as fast-flux, in which proxies are automatically constructed to host the website, algorithmic production of new URLs, etc. This method's primary flaw is that it cannot identify phishing attacks that occur at zero hour.

2.2 Reference

AUTHORS	TITLE	METHODOLOGY	PROS	CONS	INFERENCE
Arathi Krishna V, Anusree A, BlessyJose, Karthika Anilkumar	Phishing Detection using Machine Learning based URL Analysis: A Survey	classifiers Model-based Features	Create a new type of feature like Markov feature	Need large mail server and high memory requirement	In the attempt to improve URL blacklist based approaches, Phish Storm was introduced to identify potential phishing websites
Rishikesh Mahajan, IrfanSiddavatham	Phishing Website Detection using Machine Learning Algorithms	Compared multi classifier algorithm	Provide clear idea about the effective level of each classifier on phishing detection	No standard classifier	The algorithms that are used for the classification are: SVM, logistic regression, Random Forest, Bayesian Network.

Gunter Ollmann	The Phishing Guide Understanding & Preventing Phishing Attacks	Cluster of phishing Email	Fast in classification process	Less accuracy	Understanding & Preventing Phishing Attacks By:GunterOllman Director of Security Strategy IBM Internet Security System
R.Mohamad F.Thabtah L.Mccluskey	Phishing websites dataset	Evolving connectionist system	Less consuming memory	Need feed continuous	Such Dataset have been collected using our own tool, in the attached pdf document can find details of the dataset and the features in these datasets
Zou Futal Gang Yuxiang Pei Bei Pan li Li Linsen	Web Phishing detection based on graph mining	Methods based on Bag-of Words model	Build secure connection between user's mail transfer agent and mail user agent	Time consuming and huge number of features	With detecting method for phishing continually proposed and applied, the threat of web phishing has already reduced at a great extent

2.3 Problem Statement Definition

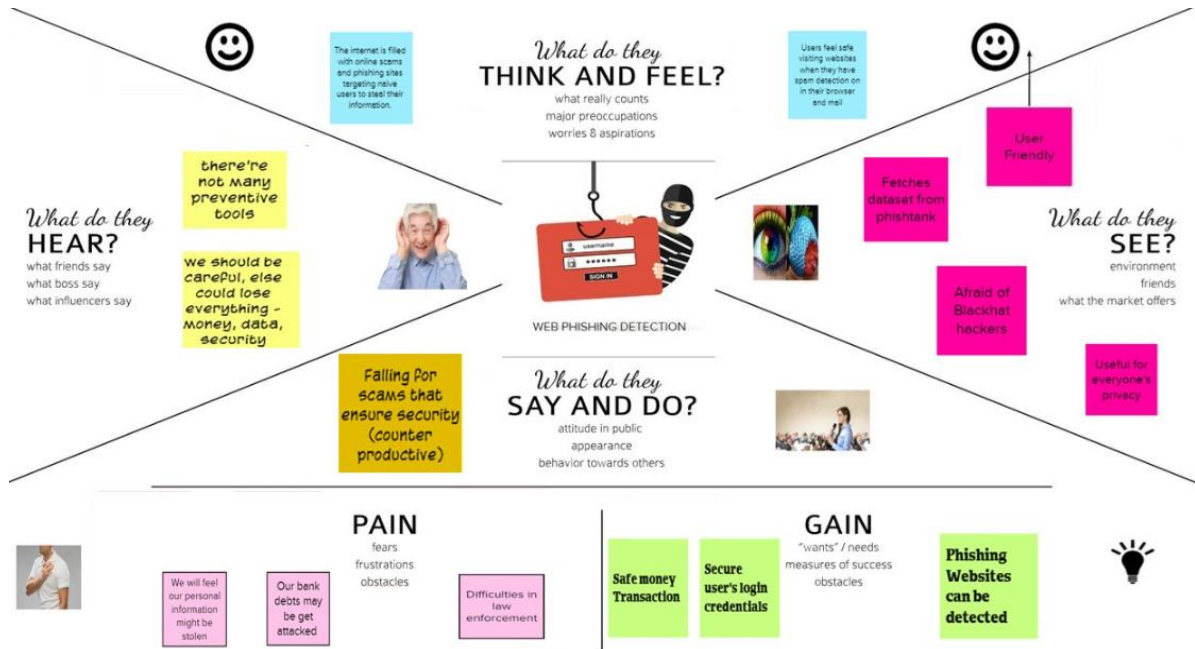
Human users' inability to recognize phishing sites allows phishing attacks to succeed. Past work in anti-phishing can be broadly divided into four categories: studies to understand why people fall for phishing attacks, strategies for teaching people not to fall for phishing attacks, user interfaces for assisting people in making better decisions about trusting email and websites, and automated tools to detect phishing. Our research outlines a method for automatically identifying phishing.

Most end users typically base their decisions only on how they feel and how they look. When a user accesses the internet, all they see is a browser's screen. After that, he or she works on a web page's command. Most phishing efforts take use of this sort of unintended chance provided by the user and trick them since the user is unconcerned with the back end procedure.

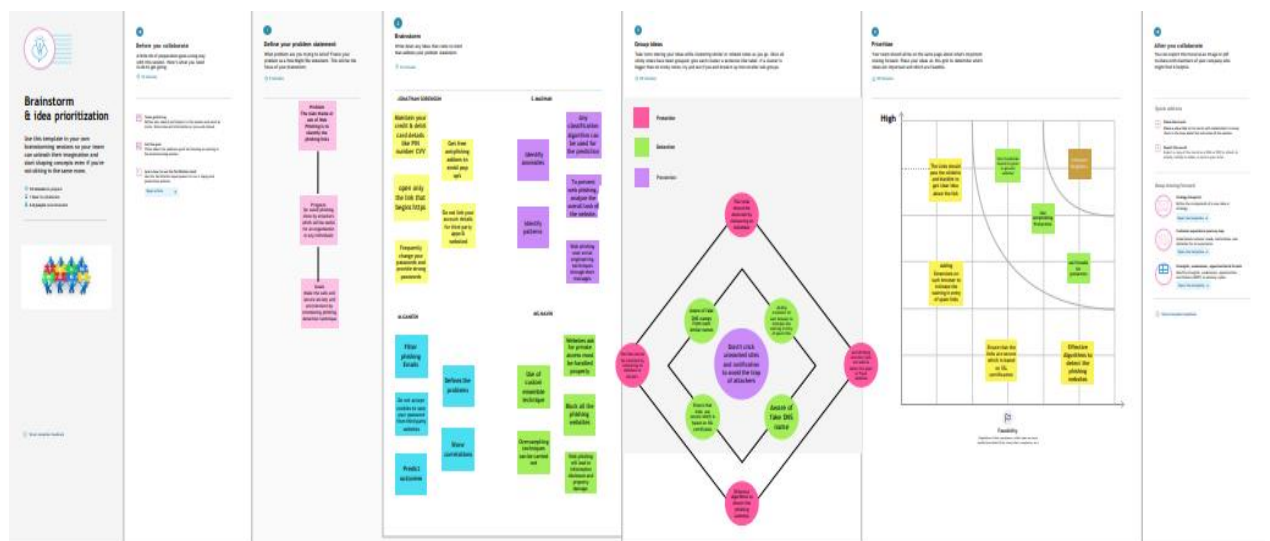
The goal of phishers is to obtain vital data, such as username, password, and bank account information. In this research, many properties of legal and phishing URLs are extracted and analyzed in order to detect phishing URLs. The algorithms used to identify phishing websites include decision trees, random forests, and support vector machines. By evaluating each algorithm's accuracy rate, false positive rate, and false negative rate

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



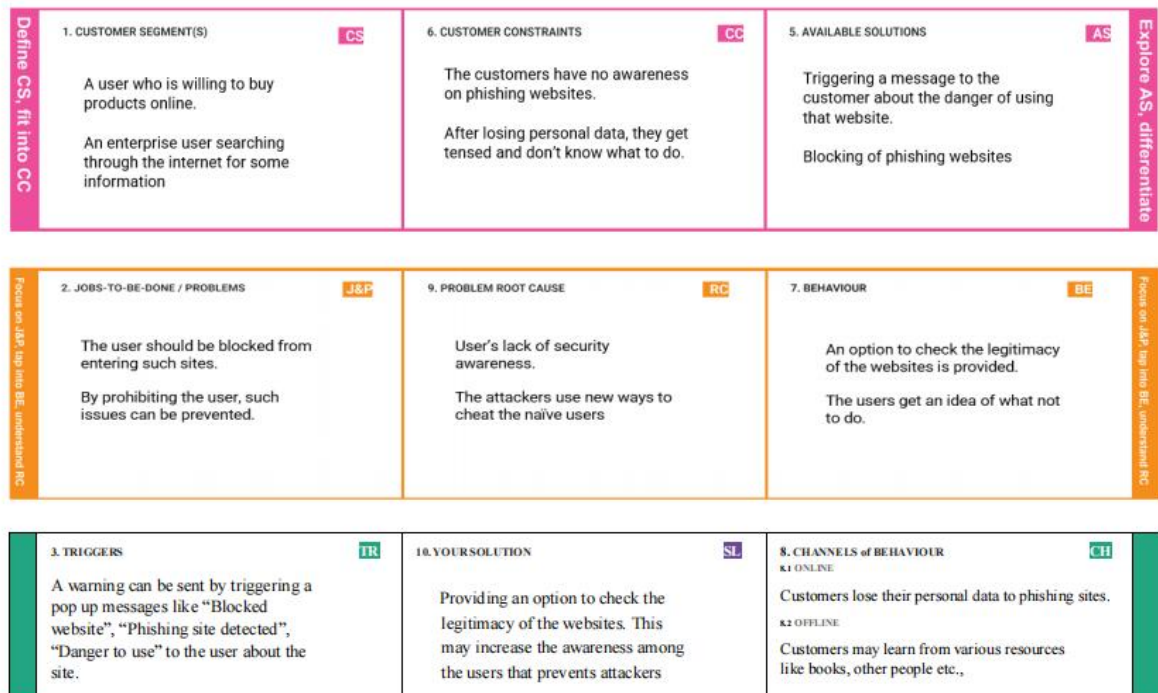
3.2 Ideation & Brainstorming



3.3 Proposed Solution

Sl.NO	PARAMETER	DESCRIPTION
1	Idea / Solution description	As a browser extension, our solution scrapes the website URL and feeds it into our machine learning model. The extension alerts the user if our model determines that the website is a phishing webs
2	Novelty / Uniqueness	There are no prior works that make advantage of the browser extension factor. As a browser extension, the user doesn't need to second-guess using a website because our plugin will
3	Social Impact / Customer Satisfaction	The user does not have to do any work because this is a fairly hands-off approach; instead, the extension will educate the user about the legitimacy of the website. Due to the fact that less information is stolen by phishing sites, customers are more satisfied as a result. when accessing a trustworthy website, with ease
4	Business Model (Revenue Model)	We suggest a two-tier system, with one tier being "free" and the other "premium." When classifying a website, the free tier would include the advertisements that appear underneath.
5	Scalability of the Solution	Since this would be a Chrome Marketplace-published browser extension, it can Everybody in the globe can access and use it. It could be scaled up because it is hosted on the IBM Cloud as needed, up and down

3.4 Problem Solution fit



4 . REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR NO	Functional Requirement	Classification
FR - 1	Fetch Electronic Mail Message	Core
FR - 2	Extract URLS	Core
FR - 3	Extract Header Information	Core
FR - 4	Classify Email	Core
FR - 5	Static or Dynamic	Core
FR - 6	Provide User Feedback	Core

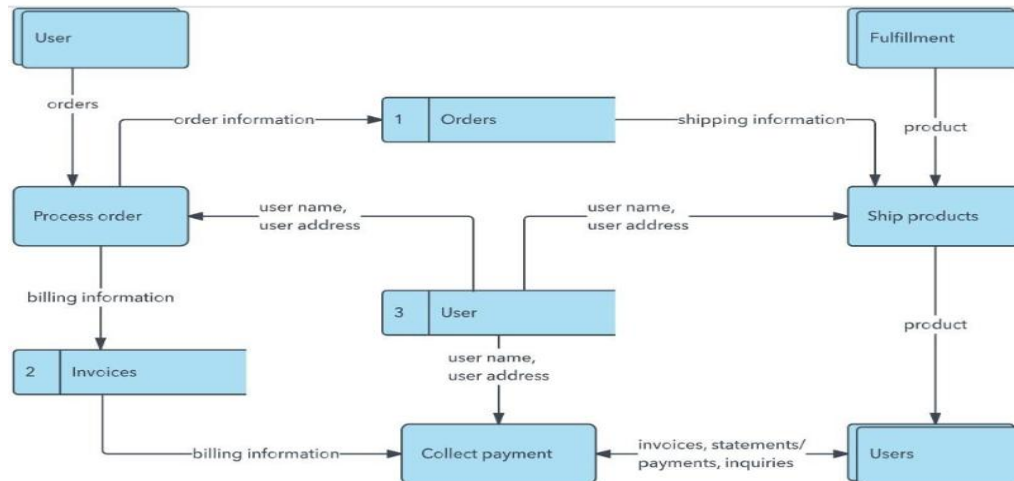
4.2 Non-Functional requirements

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

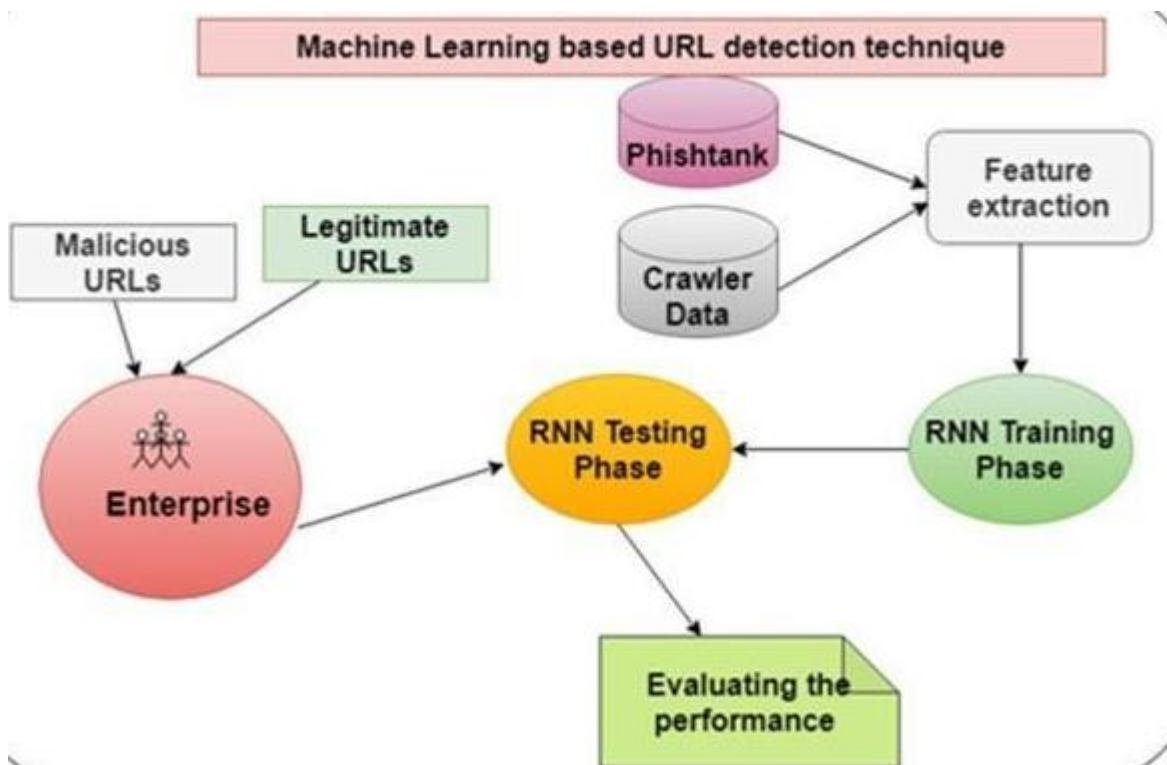
FR NO	Non Functional Requirement	Description
NFR - 1	Usability	System is easy to configure and is efficient in carrying out user tasks
NFR - 2	Availability	System is available to work as required when it is required.
NFR - 3	Reliability	System will perform the tasks it was designed to do.
NFR - 4	Performance	System will perform tasks in a fashion that complies with predetermined criteria
NFR - 5	Security	System will protect all data manipulated internally from unauthorized access and threats.
NFR - 6	Scalability	System will appropriately handle increasing and decreasing workloads.

5 . PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

Using below template the user stories for the product are derived

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirm in Any password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user i can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After i compare in case if none found on comparison then we can extract	As a User i can have comparison	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	1	High	M.Kiruthika parameshwari
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	1	Low	L.Umabharathi
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	1	High	S.Shrreenidhi
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	L.Umabharathi
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier and produces the final result.	2	High	TR.Shanmugapriya
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	M.Kiruthika parameshwari
Sprint-4	Events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	Medium	S.Shrreenidhi

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed(as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	26 Oct 2022	31 Oct 2022	20	31 Oct 2022
Sprint-2	20	6 Days	2 Nov 2022	07 Nov 2022	20	07 Nov 2022
Sprint-3	20	6 Days	09 Nov 2022	14 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days	16 Nov 2022	21 Nov 2022	20	16 Nov 2022

Velocity:

$$AV = \text{Velocity} / \text{Duration} = 35 / 7 = 5$$

$$AV = \text{Velocity} / \text{Duration} = 15 / 8 = 1.875A$$

$$V = \text{Velocity} / \text{Duration} = 25 / 8 = 3.125$$

7. CODING & SOLUTIONING

7.1 Feature 1

HTML Page

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

  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baD1411NQApFmC26EwAOH8WgZl5MYYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

  <link href="/static/styles.css" rel="stylesheet">
  <title>PHISHING URL detection</title>
</head>

<body>

<div class="container">
  <div class="row">
    <div class="form col-md" id="form1">
      <h2>PHISHING URL DETECTION</h2>

      <br>
      <form action="/" method="post">
        <input type="text" class="form_input" name="url" id="url" placeholder="Enter URL" required="" />
        <label for="url" class="form_label">URL</label>
        <button class="button" role="button">Click to Check</button>
      </form>
    </div>
  </div>
</div>
```

```
    <button class="button" role="button">Click to Check</button>
  </form>
</div>

<div class="col-md" id="form2">
  <br>
  <h6 class="right"><a href="{{ url }}" target="_blank">{{ url }}</a></h6>

  <br>
  <h3 id="prediction"></h3>
  <button class="button2" id="button2" role="button" onclick="window.open('{{url}}')" target="_blank">Still want to
  <button class="button1" id="button1" role="button" onclick="window.open('{{url}}')" target="_blank">Continue</butt
</div>
</div>
<br>
<h1></h1>
</div>

<!-- JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXAkRfj"
        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+6QL9UvVjzE3Ipu6Tp75j7Bh/kR0JkI"
        crossorigin="anonymous"></script>
```


7.2 FEATURE 2

```
pp.py 7 ...
# importing required libraries
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction
import requests

file = open("pickle/model.pkl", "rb")
gbc = pickle.load(file)
file.close()
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "kyeXQxqwmZT0TMAYGNOSZMoIhb2fMj_XkKEJvM1-iIBF"
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}

app = Flask(__name__)

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

```
pp.py 7 ...
18 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
19 API_KEY = "kyeXQxqwmZT0TMAYGNOSZMoIhb2fMj_XkKEJvM1-iIBF"
20 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
21     API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
22 mltoken = token_response.json()["access_token"]
23
24 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
25
26
27 app = Flask(__name__)
28
29 @app.route("/", methods=["GET", "POST"])
30 def index():
31     if request.method == "POST":
32         url = request.form["url"]
33         obj = FeatureExtraction(url)
34         x = np.array(obj.getFeaturesList()).reshape(1,30)
35         y_pred = gbc.predict(x)[0]
36         #1 is safe
37         #-1 is unsafe
38         y_pro_phishing = gbc.predict_proba(x)[0,0]
39         y_pro_non_phishing = gbc.predict_proba(x)[0,1]
40         # if(y_pred ==1 ):
41         pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
42         return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url )
43     return render_template("index.html", xx =-1)
44
45
46 if __name__ == "__main__":
47     app.run(debug=True)
```

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments
HomePage_TC_001	Functional	Home Page	Verify user is able to enter the URL in the form	Run the flask app in local host	1.Open our phishing website 2.Login to use the phishing services 3.Enter the link to be detected and click on predict button	https://google.com/	Result of classification will be displayed	Working as expected	Pass	Since www.google.com is a safe link, the output would display and say it is a safe link
ResultPage_TC_001	UI	Contact us page	Verify the UI elements in the form	Run the flask app in local host	1.Enter name, email and message 2.Press submit	-	An email received stating that the message has been forwarded to the team	Working as expected	Pass	Email JS is used to send automatic email
ResultPage_TC_002	Functional	Prediction result page	Verify user is able to see an alert when	Run the flask app in local host	1.Enter URL and click go		Alert of incomplete input	Working as expected	Pass	

8.2 User Acceptance Testing

1. Defect Analysis:

* This report shows the number of resolved or closed bugs at each severity.

* The Defect analysis dashboard provides an overview of product defects and inspection rates. The dashboard is made up of a number of reports that analyze defects by event code, location, and production batch.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77



2. Test Case Analysis:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5-
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9 . RESULTS

9.1 Performance Metric

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Decision Tree Model Accuracy – 97%	 <pre># Decision Tree from sklearn.tree import DecisionTreeClassifier dt = DecisionTreeClassifier() dt.fit(X_train, y_train) train_score = dt.score(X_train, y_train) test_score = dt.score(X_test, y_test) train_score, test_score (1.0, 0.990262550320982) y_pred = dt.predict(X_test) accuracy = metrics.accuracy_score(y_test, y_pred) accuracy 0.990262550320982</pre>
2.	Accuracy	Training Accuracy - Test Accuracy -	 <pre># Decision Tree from sklearn.tree import DecisionTreeClassifier dt = DecisionTreeClassifier() dt.fit(X_train, y_train) train_score = dt.score(X_train, y_train) test_score = dt.score(X_test, y_test) train_score, test_score (1.0, 0.990262550320982) y_pred = dt.predict(X_test) accuracy = metrics.accuracy_score(y_test, y_pred) accuracy 0.990262550320982</pre>

10. ADVANTAGES & DISADVANTAGES:

Phishing is the attempt to obtain a user's financial and personal information, such as credit card numbers and passwords, through electronic communication such as email and other messaging services. Attackers pose as representatives of a company and direct users to a fake website that looks like a phishing website, which is then used to gather personal data about users. A link embedded in the email can be used by attackers to trick users into downloading malware or malicious software.

To protect users from phishing attacks, numerous studies have been conducted. Firewalls, the blocking of specific domains and IP addresses, spam filtering methods, the detection of phoney websites, client-side toolbars, and user education are some of them. Both benefits and drawbacks may be seen in any of these methods now in use.

11. CONCLUSION

Using machine learning technologies, this initiative seeks to improve the detection process for phishing websites. Using the random forest approach, we had the lowest percentage of false positives and 97.14% detection accuracy. The outcome further demonstrates that classifiers perform better when more data is utilized as training data. Future phishing website detection will be more accurate thanks to the implementation of hybrid technology, which combines the blacklist approach with the random forest algorithm of machine learning.

12. FUTURE SCOPE

Future study will evaluate the effectiveness of the current finding with the use of a different method, such as deep learning, for phishing web page identification. Additionally, a web browser plug-in that can identify phishing websites and shield consumers in real time will be created based on an effective algorithm. For simple access to human life, service providers provide a variety of the quickest instruments online. Additionally, online crime such as phishing is disseminated similarly to real-world crime. However, there is no online security team protecting users from these crimes. All types of internet users can benefit greatly from an anti-phishing program. These security tools are more necessary for beginners or people with limited internet or e-commerce knowledge. Phishing's primary targets are online banking or payments. The ideal method for identifying cyber crime or e-marketing fraud is thus an automated anti-phishing technique.

13. APPENDIX

Source Code

```
1  #importing required libraries
2  from flask import Flask, request, render_template
3  import numpy as np
4  import pandas as pd
5  from sklearn import metrics
6  import warnings
7  import pickle
8  warnings.filterwarnings('ignore')
9  from feature import FeatureExtraction
10 import requests
11
12
13
14
15 file = open("pickle/model.pkl","rb")
16 gbc = pickle.load(file)
17 file.close()
18 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
19 API_KEY = "kyeXQxqwMZT0TMAYGNOSZMoIhb2fMj_XkKEJvMl-iIBF"
20 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
21 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
22 mltoken = token_response.json()["access_token"]
23
24 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
25
26
27 app = Flask(__name__)
28
29 @app.route("/", methods=["GET", "POST"])
30 def index():
31     if request.method == "POST":
```

```
app.py > ...
18 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
19 API_KEY = "kyeXQxqwMZT0TMAYGNOSZMoIhb2fMj_XkKEJvMl-iIBF"
20 token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
21 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
22 mltoken = token_response.json()["access_token"]
23
24 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
25
26
27 app = Flask(__name__)
28
29 @app.route("/", methods=["GET", "POST"])
30 def index():
31     if request.method == "POST":
32         url = request.form["url"]
33         obj = FeatureExtraction(url)
34         x = np.array(obj.getFeaturesList()).reshape(1,30)
35         y_pred =gbc.predict(x)[0]
36         #1 is safe
37         #-1 is unsafe
38         y_pro_phishing = gbc.predict_proba(x)[0,0]
39         y_pro_non_phishing = gbc.predict_proba(x)[0,1]
40         # if(y_pred ==1 ):
41         pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
42         return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url )
43     return render_template("index.html", xx =-1)
44
45
46 if __name__ == "__main__":
47     app.run(debug=True)
```



```

feature.py > ...
22 features = []
23 def __init__(self,url):
24     self.features = []
25     self.url = url
26     self.domain = ""
27     self.whois_response = ""
28     self.urlparse = ""
29     self.response = ""
30     self.soup = ""
31
32     try:
33         self.response = requests.get(url)
34         self.soup = BeautifulSoup(Response.text, 'html.parser')
35     except:
36         pass
37
38     try:
39         self.urlparse = urlparse(url)
40         self.domain = self.urlparse.netloc
41     except:
42         pass
43
44     try:
45         self.whois_response = whois.whois(self.domain)
46     except:
47         pass
48
49
50
51
52 self.features.append(self.UsingIp())
53 self.features.append(self.LongURL())

```

```

feature.py > ...
54 self.features.append(self.shortUrl())
55 self.features.append(self.symbol())
56 self.features.append(self.redirecting())
57 self.features.append(self.prefixSuffix())
58 self.features.append(self.SubDomains())
59 self.features.append(self.Hppts())
60 self.features.append(self.DomainRegLen())
61 self.features.append(self.Favicon())
62
63
64 self.features.append(self.NonStdPort())
65 self.features.append(self.HTTPSDomainURL())
66 self.features.append(self.RequestURL())
67 self.features.append(self.AnchorURL())
68 self.features.append(self.LinksInScriptTags())
69 self.features.append(self.ServerFormHandler())
70 self.features.append(self.InfoEmail())
71 self.features.append(self.AbnormalURL())
72 self.features.append(self.WebsiteForwarding())
73 self.features.append(self.StatusBarCust())
74
75 self.features.append(self.DisableRightClick())
76 self.features.append(self.UsingPopupWindow())
77 self.features.append(self.IframeRedirection())
78 self.features.append(self.AgeofDomain())
79 self.features.append(self.DNSRecording())
80 self.features.append(self.WebsiteTraffic())
81 self.features.append(self.PageRank())
82 self.features.append(self.GoogleIndex())
83 self.features.append(self.LinksPointingToPage())
84 self.features.append(self.StatsReport())
85

```

```

226         for embed in self.soup.find_all('embed', src=True):
227             dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
228             if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
229                 success = success + 1
230             i = i+1
231
232         for iframe in self.soup.find_all('iframe', src=True):
233             dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
234             if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
235                 success = success + 1
236             i = i+1
237
238         try:
239             percentage = success/float(i) * 100
240             if percentage < 22.0:
241                 return 1
242             elif ((percentage >= 22.0) and (percentage < 61.0)):
243                 return 0
244             else:
245                 return -1
246         except:
247             return 0
248     except:
249         return -1
250
251     # 14. AnchorURL
252     def AnchorURL(self):
253         try:
254             i,unsafe = 0,0
255             for a in self.soup.find_all('a', href=True):
256                 if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (url
257                     unsafe = unsafe + 1

```

```

257         unsafe = unsafe + 1
258         i = i + 1
259
260         try:
261             percentage = unsafe / float(i) * 100
262             if percentage < 31.0:
263                 return 1
264             elif ((percentage >= 31.0) and (percentage < 67.0)):
265                 return 0
266             else:
267                 return -1
268         except:
269             return -1
270
271     except:
272         return -1
273
274     # 15. LinksInScriptTags
275     def LinksInScriptTags(self):
276         try:
277             i,success = 0,0
278
279             for link in self.soup.find_all('link', href=True):
280                 dots = [x.start(0) for x in re.finditer('\.', link['href'])]
281                 if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
282                     success = success + 1
283                 i = i+1
284
285             for script in self.soup.find_all('script', src=True):
286                 dots = [x.start(0) for x in re.finditer('\.', script['src'])]
287                 if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
288                     success = success + 1

```



```

feature.py > ...
392 # 24. AgeofDomain
393 def AgeofDomain(self):
394     try:
395         creation_date = self.whois_response.creation_date
396         try:
397             if(len(creation_date)):
398                 creation_date = creation_date[0]
399         except:
400             pass
401
402         today = date.today()
403         age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
404         if age >=6:
405             return 1
406         return -1
407     except:
408         return -1
409
410 # 25. DNSRecording
411 def DNSRecording(self):
412     try:
413         creation_date = self.whois_response.creation_date
414         try:
415             if(len(creation_date)):
416                 creation_date = creation_date[0]
417         except:
418             pass
419
420         today = date.today()
421         age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
422         if age >=6:

```

```

473         return -1
474
475 # 30. StatsReport
476 def StatsReport(self):
477     try:
478         url_match = re.search(
479             'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', urlsafe_b6
480             ip_address = socket.gethostbyname(self.domain)
481             ip_match = re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158
482             '107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|
483             '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|14
484             '216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|2
485             '34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.2
486             '216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82
487
488         if url_match:
489             return -1
490         elif ip_match:
491             return -1
492         return 1
493     except:
494         return 1
495
496 def getFeaturesList(self):
497     return self.features

```

GitHub & Project Demo Link

GitHub Link

<https://github.com/IBM-EPBL/IBM-Project-43325-1660715910>

Demo Link

https://drive.google.com/drive/folders/1CBrGres0gTiw_3x7h2Xld31SgoAJGslj?usp=share_link