IBM
NALAIYA THIRAN

PROJECT REPORT
ON
WEB PHISHING DETECTION

TEAM ID: PNT2022TMID32563

UNIVERSITY COLLEGE OF ENGINEERING (BIT CAMPUS)
ANNA UNIVERSITY

Team Members

SUBASRI R
SWETHA A
SWETHA D
SADHANA M

TABLE OF CONTENTS

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: 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 11

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 preprocessed 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 realtime 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

PAPER 2.1: PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning.

Authors: Ankit Kumar Jain **&** B.B.Gupta

Abstract:

Today, phishing is one of the most serious cyber-security threat in which attackers steal sensitive information such as personal identification number(PIN), credit card details, login, password, etc., from Internet users. In this paper, we proposed a machine learning based anti-phishing system (i.e., named as PHISH- SAFE) based on Uniform Resource Locator (URL) features. To evaluate the performance of our proposed system, we have taken 14 features from URL to detect a website as a phishing or non-phishing. The proposed system is trained using more than 33,000 phishing and legitimate URLs with SVM and Naïve Bayes classifiers.

Our experiment results show more than 90% accuracy in detecting phishing websites using SVM classifier.

PAPER 2.2: Detection of URL based phishing attacks using machine learning

Authors: Ms. Sophiya. Shikalgar, Dr. S. D. Sawarkar, Mrs. Swati Narwane

Abstract:

A fraud effort to get sensitive and personal information like password, username, and bank details like credit / debit card details by masking as a reliable organization in electronic communication. It most of the time redirects the users to similar looking website as legitimate website. The phishing website will appear same as the legitimate website and directs the user to a page to enter personal details of the user on the fake website. The system administration is very important these days as any failure can be detected and solved instantly. The system administration also need to define rules and set firewall settings to avoid phishing attacks through URL. Researchers have been studying various machine learning algorithm in lines to predict and avoid phishing attacks. Through machine learning algorithms one can improve the accuracy of the prediction. The machine learning, no one algorithm works best for every problem, and it's especially relevant for supervised learning. Using a single machine learning algorithm will give us good accuracy to predict the phishing attacks but to get better accuracy we need something more. The proposed

system predicts the URL based phishing attacks with maximum accuracy. We shall talk about various machine learning, the algorithm which can help in decision making and prediction. We shall use more than one algorithm to get better accuracy of prediction. The algorithms namely the Naive Bayes and Random forest are used in the proposed system to detect URL based phishing attacks. The hybrid algorithm approach by combining.

PAPER 2.3: An Ideal Approach for Detection and Prevention of Phishing Attacks
Authors: Narendra.M & Chaithali shah
Abstract:

In this paper, we propose a phishing detection and prevention approach combining URL-based and Webpage similarity based detection. URL-based phishing detection involves extraction of actual URL (to which the website is actually directed) and the visual URL (which is visible to the user). LinkGuard Algorithm is used to analyze the two URLs and finally depending on the result produced by the algorithm the procedure proceeds to the next phase. If phishing is not detected or Phishing possibility is predicted in URL-based detection, the algorithm proceeds to the visual similarity based detection. A novel technique to visually compare a suspicious page with the legitimate one is presented.

PAPER 2.4: Phishing website detection based on effective machine learning approach
Authors: Lokesh.G & Gowtham.B
Abstract:
Phishing a form of cyber-attack, which has an adverse effect on people where the user is directed to fake websites and duped to reveal their sensitive and personal information which includes passwords of accounts, bank details, atm pin-card details etc. Hence protecting sensitive information from malwares or web phishing is difficult. Machine learning is a study of data analysis and scientific study of algorithms, which has shown results in recent times in opposing phishing pages when distinguished with visualization, legal solutions, including awareness workshops and classic anti-phishing approaches. This paper examines the applicability of ML techniques in identifying phishing attacks and report their positives and negatives. In specific, there are many ML algorithms that have been explored to declare the appropriate choice that serve as anti-phishing tools. We have designed a Phishing Classification system which extracts features that are meant to defeat common phishing detection

approaches. We also make use of numeric representation along with the comparative study of classical machine learning techniques like Random Forest, K nearest neighbours, Decision Tree, Linear SVC classifier, One class SVM classifier and wrapper-based features selection which contains the metadata of URLs and use the information to determine if a website is legitimate or not.

PAPER 2.5: Machine Learning and Deep Learning Based Phishing Websites Detection: The Current Gaps and Next Directions

Authors: Kibreab Adane & Berhanu Beyene

Abstract:

There are many phishing websites detection techniques in literature, namely white-listing, black-listing, visual-similarity, heuristic-based, and others.

However, detecting zero-hour or newly designed phishing website attacks is an inherent property of machine learning and deep

learning techniques. By considering a promising solution of machine learning and deep learning techniques, researchers have made a great deal of effort to tackle the this problem, which persists due to attackers constantly devising novel strategies to exploit vulnerability or gaps in existing anti-phishing measures. In this study, an extensive effort has been made to rigorously review recent studies focusing on Machine Learning and Deep Learning Based Phishing Websites Detection to excavate the root cause of the aforementioned problems and offer suitable solutions. The study followed the significant criterion to search, download, and screen relevant studies, then to evaluate criterion-based selected studies. The findings show that significant research gaps are available in the rigorously reviewed studies. These gaps are mainlyrelated to imbalanced dataset usage, improper selection of dataset source(s), the unjustified reason for using specific train-test dataset split ratio, scientific disputes on website features inclusion and exclusion, lack of universal consensus on phishing website lifespans and on what is defining a small dataset size, and run-time analysis issues.

PAPER 2.6: Detection of phishing websites using an efficient feature-based machine learning framework.

Authors: Royhu Srinivas rao & sathvik

Abstract: In this paper, we propose a novel classification model, based on heuristic features that are extracted from URL, source code, and third-party services to overcome the disadvantages of existing anti-phishing techniques. Our model has been evaluated using eight different machine learning algorithms and out of which, the Random Forest (RF) algorithm performed the best with an accuracy of 99.31%. The experiments were repeated with different (orthogonal and oblique) random forest classifiers to find the best classifier for the phishing website detection. Principal component analysis Random Forest (PCA-RF) performed the best out of all oblique Random Forests (oRFs) with an accuracy of 99.55%. We have also tested our model with the third-party-based features and without third-party-based features to determine the effectiveness of third-party services in the classification of suspicious websites. We also compared our results with the baseline models (CANTINA and CANTINA+).

Our proposed technique outperformed these methods and also detected zero-day.

2.1 EXISTING PROBLEM

Phishing is a cyber attack that uses email as its method of attack. The objective is for the recipient to believe the message is legitimate and to click a link, open an attachment. Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. Attachments from phishing emails can contain malware that once opened can leave the door open to the attacker to perform malicious behavior from the user's computer. Due to their low bar of skill required to launch, phishing is a popular choice for cyber criminals. Many of them use phishing kits, which include all the technical materials needed to launch a phishing campaign.More advanced phishing methods like spoofing (pretending to send emails from a legitimate source), spear phishing (personalizing emails to target specific people), and whaling (targeting high-level executives) remain popular and are even harder to detect by eye alone. Phishing is the most popular attack vector for criminals

and has grown 65% in the last year, according to Retruser. Data shield is here to explain phishing, how attacks have affected businesses, how this form of cybercrime is growing, and how to defend against them.Phishing targets individuals and private citizens each day. Additionally, cyber criminals will target businesses.Business email compromise (BEC) scams accounted for over $12 million in losses last year, according to Retruster.Contrary to popular belief, phishing attacks are being launched on small and medium-sized businesses with shocking regularity. And while the most common industries targeted are Software-as-a-Service and Webmail organizations, social media and e-commerce industries also top the list.Beyond monetary damages, businesses who are breached lose public trust and must work to secure their databases. Many companies are required to notify their customers of a breach, pay regulatory fines, and lose customers as a result. Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced.

2.2 REFERENCES

[1]Phishing Website Detection using Machine Learning Algorithms
Rishikesh Mahajan
MTECH Information Technology
K.J. Somaiya College of Engineering, Mumbai - 77
Irfan Siddavatam
Professor, Dept. Information Technology
K.J. Somaiya College of Engineering, Mumbai - 77

[2] Detecting Phishing Websites Using Machine Learning Aniket Garje1 , Namrata Tanwani1 , Sammed Kandale1 , Twinkle Zope1 , Prof. Sandeep Gore2
1 UG Students,2Assistant Professors, Computer Engineering Department, G H Raisoni College of Engineering and Management, Pune

[3]A Survey of Phishing Website Detection Systems Prachit Raut1, Harshal Vengurlekar2, Rishikesh Shete3 1,2,3Department of Computer Engineering, Vasantdada Patil Pratishthan's College of Engineering and Visual Arts, Mumbai, Maharashtra, India

[4]A Literature Survey of Phishing Attack Technique Pratik Patil1 , Prof. P.R. Devale2 M Tech Student, Information Technology, BVUCOE, Pune, India1 Professor, Information Technology, BVUCOE, Pune, India

[5]A Survey of URL-based Phishing Detection Eint Sandi Aung†a) Chaw Thet Zan†b) and Hayato YAMANA†c) Department of Computer Science and Communication Engineering, Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, 159-8555, Japan. E-mail : a)

eintsandiaung@toki.waseda.jp, b) chawthetzan@fuji.waseda.jp, c) yamana@waseda.jp

2.3 PROBLEM STATEMENT DEFINITION

Mr. Naveen was a Rich Man. He ordered a Car in online and wanted to do online Payment, While Making this payment in e-banking website, he faced the phishing attacks like money loss, his bank account can be hacked and bank details can be hacked.

- Naveen wants to know Legitimate Website for online transaction.
- He needs to know the result immediately.
- This problem is usually faced by most of the people.

Who does the problem affect?	Persons who do e-banking transactions (Online transaction).
What are the boundaries of the problem?	People who purchase products in online and face issues in illegal websites.
What is the issue?	While making the net banking transaction in Phishing websites which is not a legitimate one, the people may loss their money or their sensitive data such as username, password, and other credit card details may be stolen.
When does the issue occur?	During the Online Payment or e-banking transactions.

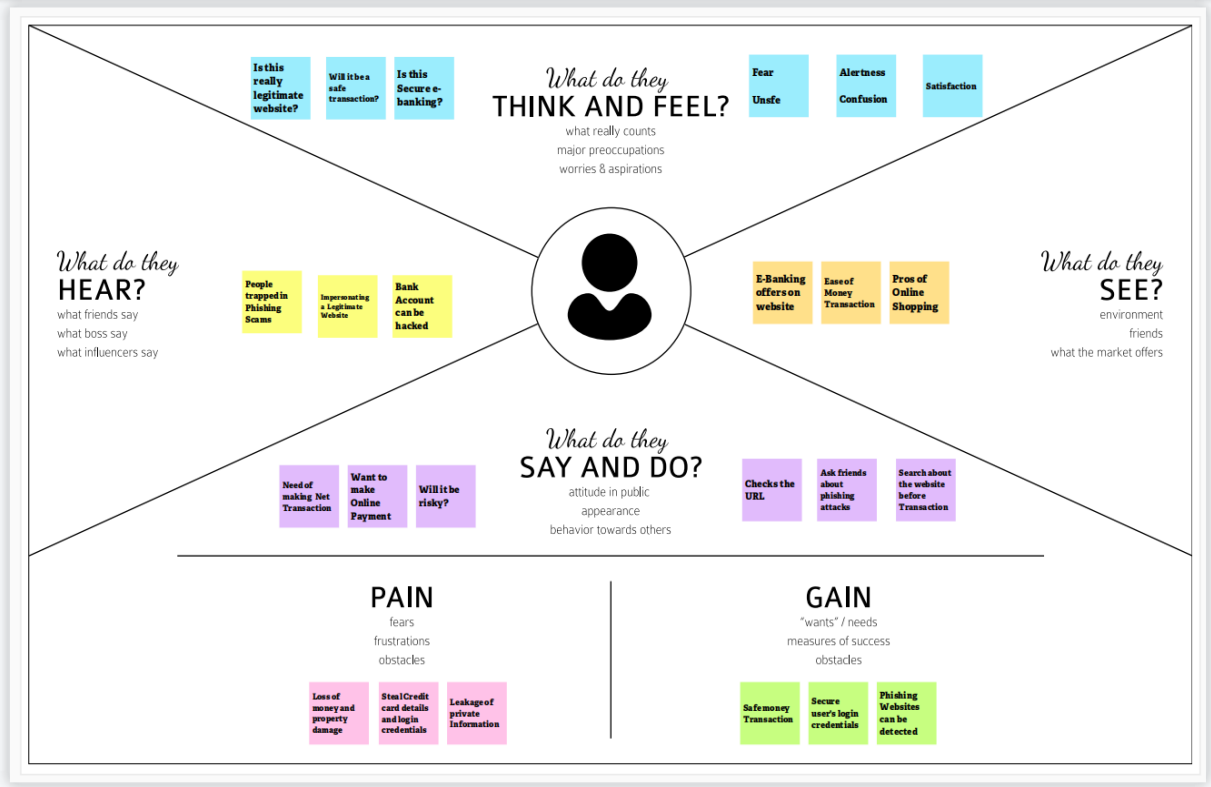
Where does the issue occur?	These issue occurs in money transaction Websites.
Why is it important that we fix the problem?	• It is required for the Safe Money Transaction and to maintain secure user's sensitive data. • It is important to reduce the loss of people's money and to reduce phishing attacks.

What solution to solve this issue?	• A system is introduced to detect the Phishing URLs and to identify fake vs real URLs. • It alerts when user may access the Phishing URLS. • This system allows only Legitimate URLs.
What methodology used to solve the Issue?	Machine learning Algorithms are used to detect Phishing websites

CHAPTER-3

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 IDEATION AND BRAINSTORMING

3.3 PROPOSED SOLUTION

Project Design Phase-I

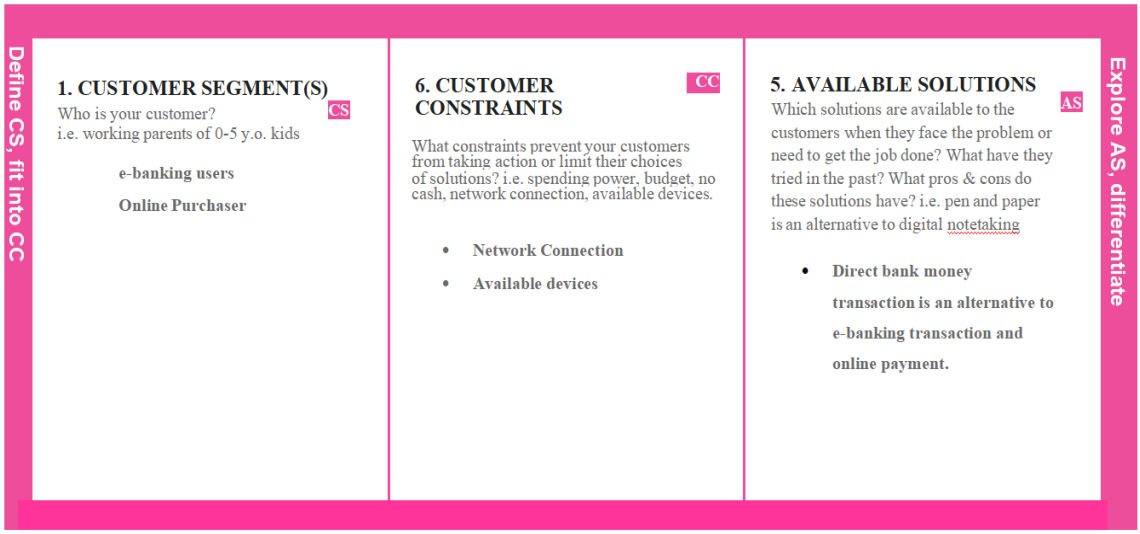
Project team will fill the following information in proposed solution template.

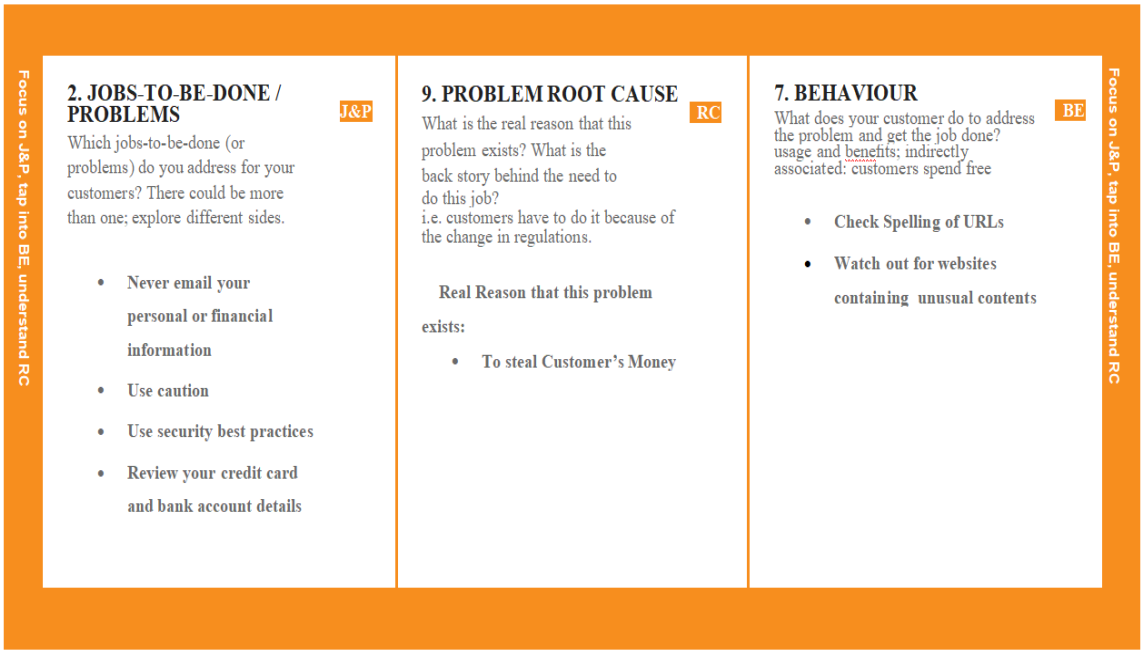
S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	There is a lot of people who purchase products in online and make payments through e-banking. E-banking websites ask users to provide sensitive data such as user name, passwords and other bank details. Some websites are Phishing websites which steal the user's data for illegal use. Here these phishing websites can be detected and allows only the Legitimate websites which is legal e- banking websites.
2.	Idea / Solution description	In machine learning algorithm, the user will find the Legitimate websites for their transaction from this project.

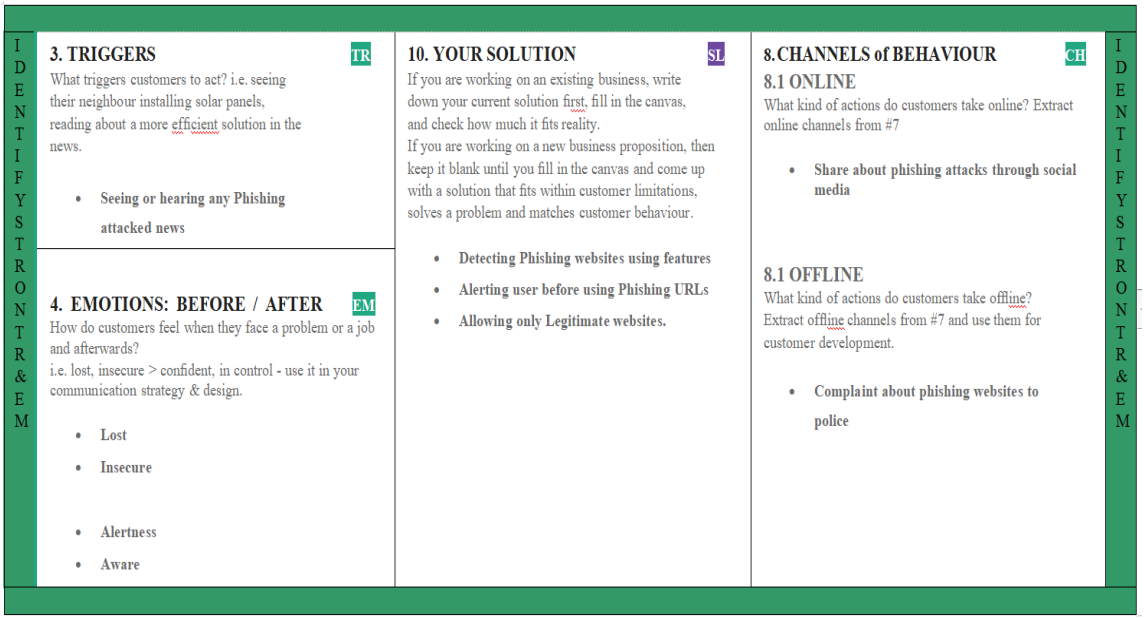
3.	Novelty / Uniqueness	• We will create a model to detect thephishing websites. • Our model will recognize fake vsreal URLs. • In our model, the website security can be tested, Alert warning for fraudulent websites.
4.	Social Impact / Customer Satisfaction	From the analysis of data, it's very clear that it reduces all the fraud or phishing websites done at the time of e-banking transaction.
5.	Business Model (Revenue Model)	Our project can be used by many E-commerce enterprises I order to make the online transaction process secure.
6.	Scalability of the Solution	It's possible to make changes to software, which can accept new testing data and should also take part in training data and predict accordingly. In future prediction, module can be more improved and integrated.

3.4 PROBLEM SOLUTION FIT

Project Design Phase-I - Solution Fit Template







CHAPTER-4

1. REQUIREMENT ANALYSIS

Project Design Phase-II

Solution Requirements (Functional & Non-functional)

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
FR-2	User Confirmation	Confirmation via Email
FR-3	User Profile	Filling the Profile Page after logged in
FR-4	Uploading Dataset(URL)	The URLs are to be Uploaded
FR-5	Requesting Solution	Uploading URLs is compared with the Pre-defined Model and solution is generated

FR-6	Displaying Solution	The Solution is in pop-up message which contains the alert for Phishing URLs

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 System allows the user to perform the tasks easily,efficiently and effectively
NFR-2	**Security**	Assuring all data inside this project will be protected against Phishing Attacks or unauthorized access
NFR-3	**Reliability**	It takes some time to recover from any failure due to running in single server
NFR-4	**Performance**	Response time is fast
NFR-5	**Availability**	The system will be available up to 96% of time
NFR-6	**Scalability**	The system is scalable

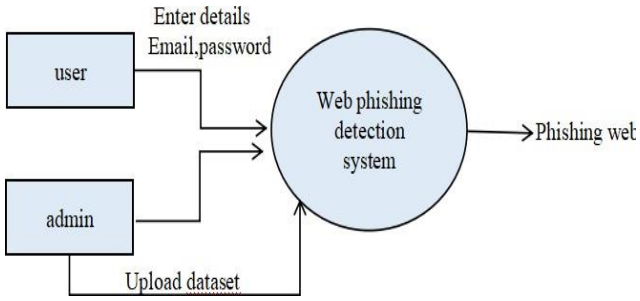
CHAPTER-5

5.PROJECT DESIGN

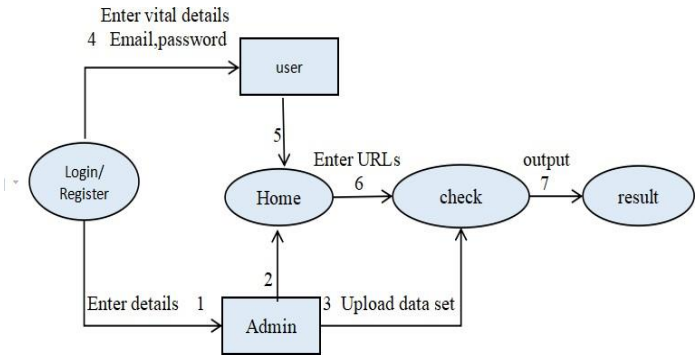
5.1 DATA FLOW DIAGRAM

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

0th LEVEL DFD :



1st LEVEL DFD



0th Level DFD Data flow

1. Admin can upload legitimate and phishing websites dataset.
2. Upload datasets for phishing detection.
3. User enter the system by Email and password.
4. After URLs entered for detection.

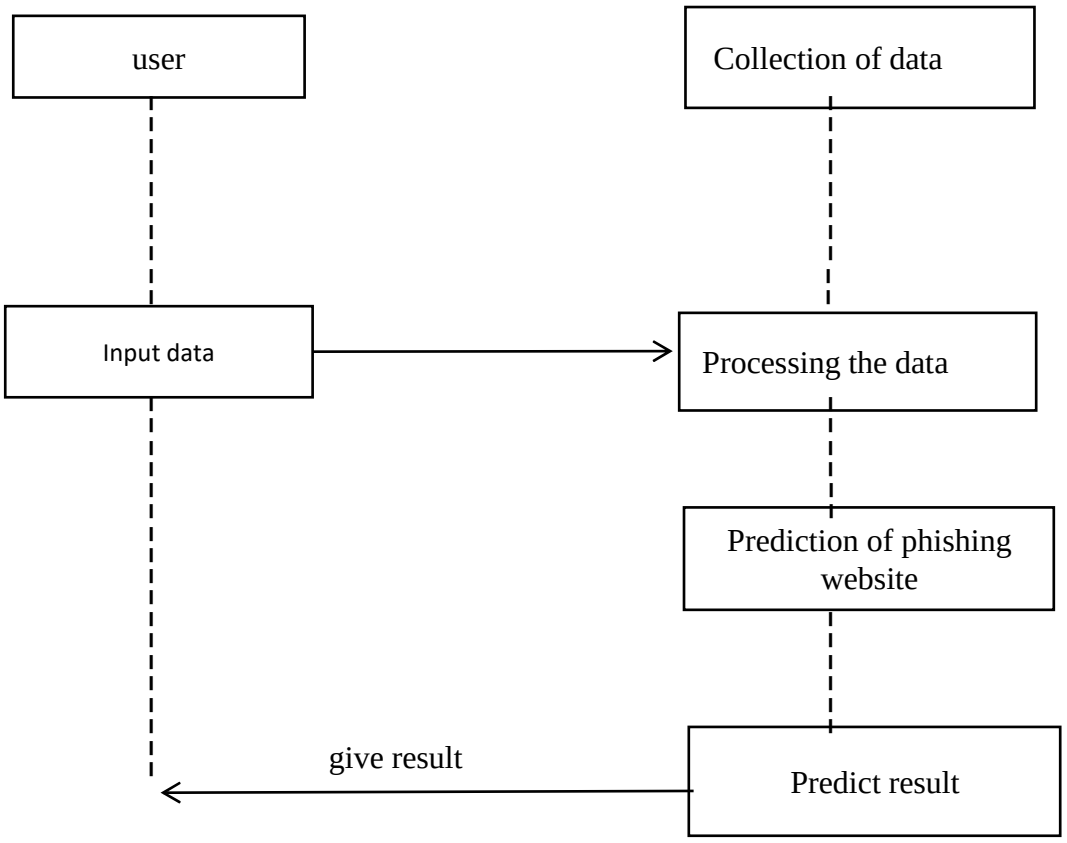
1st Level DFD Data flow:

1. Admin enters the home by logging in and upload the datasets.
2. User open the login/Register page from Homepage.
3. Admin uploading the datasets.
4. User enters email,password for Registering.
5. User is redirected to the Home once they Login.
6. User enter URLs to check from homepage.
7. Output will be displayed in the result page.

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

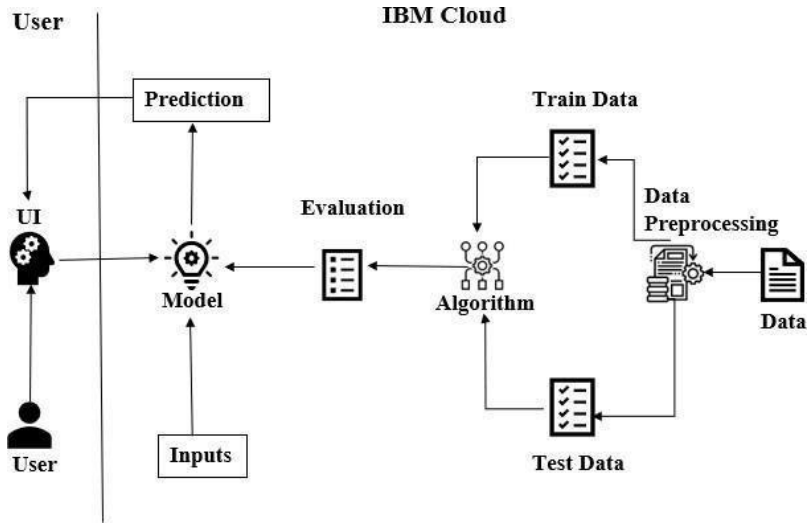
PROJECT DESIGN PHASE-1

Solution Architecture



TECHNICAL ARCHITECTURE

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User interact with our application through webUser Interface.	HTML, CSS and Python flask.
2.	Application Logic-1-Login.	When the user click on the login button , he/she isdirected to login page, if they are registered already.	HTML ,CSS, Python flask.
3.	Application Logic-Registration	When the user click on the Register button , he/sheis directed to Register page for further process.	HTML,CSS, Python flask.
4.	Application Logic-Credibility details	After Logged in , when the user click on the credibility details form button,he/she directed to the form page to enter the details of applicant forprediction.	Front end-HTML ,CSS , MySQL, Python flask Back end-Python
5.	Database	Data type - String ,Numeric.	MySQL.
6.	Cloud Database	Database Service on Cloud	IBM.
7.	File Storage	File storage requirements	NIL
8.	External API-1	Purpose of External API used in the application	NIL
9.	External API-2	Purpose of External API used in the application	Aadhar API

10.	Machine Learning Model	Get the data from the user and predict the data tested and trained dataset models	Data Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / CloudLocal Server Configuration: Cloud Server Configuration :	NIL

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	International Business Machines.	Cloud.
2.	Security Implementations	Access permission for login page usingCAPTCHA	Encryptions.
3.	Scalable Architecture	The key of Three tier architecture is improvingscalability.	Three Tier architecture.
4.	Availability	Load balancer or ADC is the key component that ensures high availability by sending request.	Load balancer.
5.	Performance	The system should be able to handle large numberof users at the time	Load balancer.

5.3 User Stories

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

User Type	Functional Requirement (Epic)	USN	User Story / Task	Acceptance criteria	Priority	Release
user	HomePage	USN - 1	Web phishing detection description	view/ access my Home page.	Low	Sprint - 3
		USN - 2	Information about Test Vitals required for Detection		Low	Sprint - 3
Admin	Admin page	USN - 3	Uploading the datasets	Access datasets	High	Sprint - 2
	User Registration	USN - 4	Enters Mail ID and other personal detailsrequired for Registering.	Successful register using my email id.	Medium	Sprint - 2
	User Login	USN - 5	Uses Mail ID and Password for login	Successful logged in.	Medium	Sprint - 2
	Test Vitals Form	USN - 6	Test urls should be entered for prediction	access the test vitals	High	Sprint - 1

User Type	Functional Requirement (Epic)	USN	User Story / Task	Acceptance criteria	Priority	Release
	Result	USN - 7	Results will be displayed.	Got result	High	Sprint - 1
		USN - 8	If Phishing – shows warningto avoid the website If not phishing – suggesting touse.	I got useful information	Low	Sprint - 4

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	International Business Machines.	Cloud.
2.	Security Implementations	Access permission for login page usingCAPTCHA	Encryptions.
3.	Scalable Architecture	The key of Three tier architecture is improvingscalability.	Three Tier architecture.
4.	Availability	Load balancer or ADC is the key component that ensures high availability by sending request.	Load balancer.
5.	Performance	The system should be able to handle large numberof users at the time	Load balancer.

CHAPTER-6

6. PROJECT PLANNING & SCHEDULING

6. 1 Sprint Planning, Delivery Schedule & Estimation

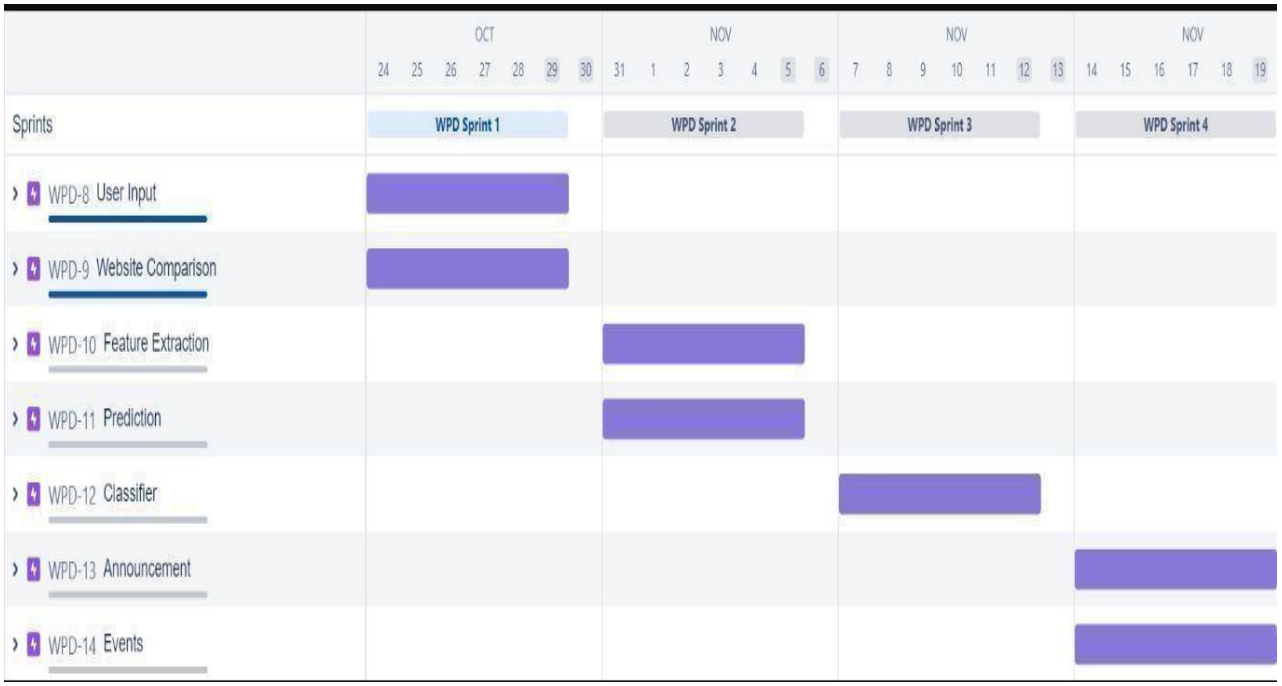
Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Home Page	USN - 1	Home Page contains Registration and login tab. Information about Web Phishing Detection.	5	Medium	Subasri R, Swetha A
Sprint-1	User Registration	USN - 2	Enter Mail ID, Username and other Bank Account details required for Registration.	8	Medium	Subasri R, Swetha A
Sprint -1	User Login	USN - 3	Uses Mail ID and Password for login.	7	Medium	Subasri R, Swetha A
Sprint-2	Test URL	USN- 4	Test URLs will be Uploaded for detection.	10	High	Sadhana M, Swetha D
Sprint-3	Detection	USN- 5	As a admin, we can use various ML classifier model for the accurate result for the detection of URL	10	High	Subasri R, Swetha A, Sadhana M, Swetha D
Sprint - 4	Result	USN - 6	• If the Phishing website is detected, the alert message is displayed in user interface. • If the detected website is Legitimate, then User is allowed to use this website.	10	High	Subasri R, Swetha A, Sadhana M, Swetha D

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	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	10	6 Days	07 Nov 2022	12 Nov 2022	10	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

6.3 Reports from JIRA



CHAPTER-7

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

7.1 Feature 1

app.py

```
import pickle
from flask import Flask, jsonify, render_template, request,redirect,url_for,request

#importing the inputScript file used to analyze the URL import inputScript
import inputScript
import numpy as np
import sklearn
```

```
from flask import Flask, render_template, request, redirect, url_for, session

from flask_mysqldb import MySQL

import MySQLdb.cursors

import re
app = Flask(__name__)

app.secret_key = 'web'

app.config['MYSQL_HOST'] = 'localhost'

app.config['MYSQL_USER'] = 'root'

app.config['MYSQL_PASSWORD'] = 'Suba@10'

app.config['MYSQL_DB'] = 'webdb'

app.config['MYSQL_CURSORCLASS'] = 'DictCursor'



mysql = MySQL(app)

#load model

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

@app.route('/')
@app.route('/homepage')
def homepage():
```

```
    return render_template('homepage.html')


@app.route('/flask/login', methods =['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in
     request.form:
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM signup WHERE email = % s AND
     password = % s', (email, password, ))
        user = cursor.fetchone()
        if user:
            session['loggedin'] = True
            session['name'] = user['name']
            session['email'] = user['email']
            mesage = 'Logged in successfully !'
            return render_template('predictform.html', msg = msg)
        else:
            mesage = 'Please enter correct email / password !'
    return render_template('login.html', msg = msg)


@app.route('/flask/reg', methods =['GET', 'POST'])

def register():
    msg = ''

    if request.method == 'POST' and 'name' in request.form  and 'email' in
```

```
request.form and 'password' in request.form:
    name = request.form['name']
    email = request.form['email']
    password = request.form['password']
    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('SELECT * FROM signup WHERE email = % s', (email, ))
    account = cursor.fetchone()
    if account:
        msg = 'Account already exists !'
    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
        msg = 'Invalid email address !'
    elif not re.match(r'[A-Za-z0-9]+', name):
        msg = 'Username must contain only characters and numbers !'
    elif not name or not password or not email:
        msg = 'Please fill out the form!'
    else:
```

```
            cursor.execute('INSERT INTO signup
        VALUES(%s,%s,%s)',(name,email,password,))

            mysql.connection.commit()

            msg = 'You have successfully registered !'

            return redirect(url_for('login'))

    elif request.method =='POST':
            msg='Please fill out the form!'

    return render_template('reg.html', msg = msg)

@app.route('/flask/predictform', methods =['GET', 'POST'])
def predictform():
    return render_template('predictform.html')

@app.route('/flask/about', methods =['GET', 'POST'])
def about():
    return render_template('about.html')

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('name', None)
    session.pop('email', None)
    return redirect(url_for('homepage'))

#Redirects to the page to give the user iput URL.
```

```
@app.route('/flask/result')
def result():
    return render_template('result.html')
ans=""
bns=""

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict', methods=['POST'])
def y_predict():

    '''
    For rendering results on HTML GUI
    '''

    url = request.form['url']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction [0]
    if(output==1):
        pred="Your are safe!! This is a Legitimate Website."
        return render_template('result.html',ans=pred)

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



#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.y_predict ( [np.array(list(data.values()))])

    output = prediction[0]
    return jsonify (output)

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

7.2 Feature 2

InputScript.py

```
import pickle
from flask import Flask, jsonify, render_template, request,redirect,url_for,request

#importing the inputScript file used to analyze the URL import inputScript
import inputScript
import numpy as np
import sklearn

from flask import Flask, render_template, request, redirect, url_for, session

from flask_mysqldb import MySQL
```

```
import MySQLdb.cursors

import re
app = Flask(__name__)

app.secret_key = 'web'

app.config['MYSQL_HOST'] = 'localhost'

app.config['MYSQL_USER'] = 'root'

app.config['MYSQL_PASSWORD'] = 'Suba@10'

app.config['MYSQL_DB'] = 'webdb'

app.config['MYSQL_CURSORCLASS'] = 'DictCursor'


mysql = MySQL(app)

#load model

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

@app.route('/')
@app.route('/homepage')
def homepage():
    return render_template('homepage.html')


@app.route('/flask/login', methods =['GET', 'POST'])
```

```
def login():
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in
     request.form:
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM signup WHERE email = % s AND
     password = % s', (email, password, ))
        user = cursor.fetchone()
        if user:
            session['loggedin'] = True
            session['name'] = user['name']
            session['email'] = user['email']
            mesage = 'Logged in successfully !'
            return render_template('predictform.html', msg = msg)
        else:
            mesage = 'Please enter correct email / password !'
    return render_template('login.html', msg = msg)


@app.route('/flask/reg', methods =['GET', 'POST'])

def register():
    msg = ''

    if request.method == 'POST' and 'name' in request.form  and 'email' in
     request.form and 'password' in request.form:

        name = request.form['name']
```

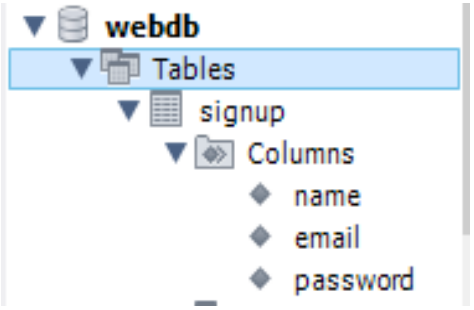
```
    email = request.form['email']

    password = request.form['password']

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    cursor.execute('SELECT * FROM signup WHERE email = % s', (email, ))

    account = cursor.fetchone()

    if account:

        msg = 'Account already exists !'

    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

        msg = 'Invalid email address !'

    elif not re.match(r'[A-Za-z0-9]+', name):

        msg = 'Username must contain only characters and numbers !'

    elif not name or not password or not email:

        msg = 'Please fill out the form!'

    else:

        cursor.execute('INSERT INTO signup
VALUES(%s,%s,%s)',(name,email,password,))
```

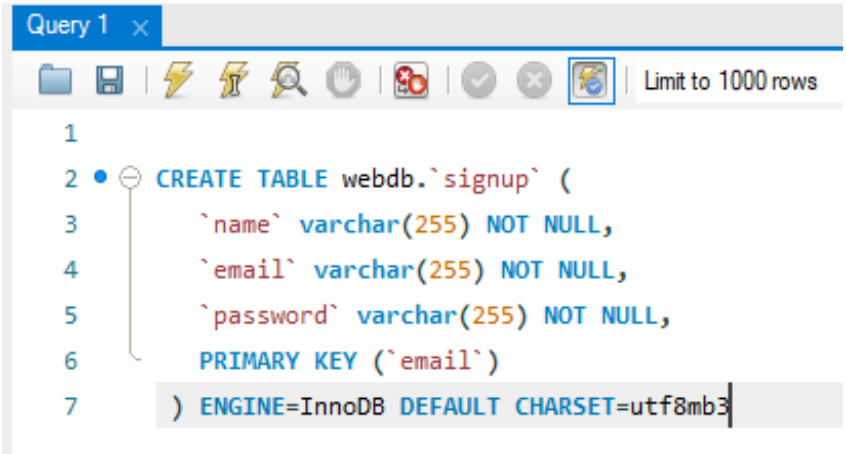
```
            mysql.connection.commit()

            msg = 'You have successfully registered !'

            return redirect(url_for('login'))

    elif request.method =='POST':
            msg='Please fill out the form!'

    return render_template('reg.html', msg = msg)

@app.route('/flask/predictform', methods =['GET', 'POST'])
def predictform():
    return render_template('predictform.html')

@app.route('/flask/about', methods =['GET', 'POST'])
def about():
    return render_template('about.html')

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('name', None)
    session.pop('email', None)
    return redirect(url_for('homepage'))

#Redirects to the page to give the user iput URL.
@app.route('/flask/result')
def result():
    return render_template('result.html')
ans=""
```

```
bns=""

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict', methods=['POST'])
def y_predict():

    '''
    For rendering results on HTML GUI
    '''

    url = request.form['url']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction [0]
    if(output==1):
        pred="Your are safe!! This is a Legitimate Website."
        return render_template('result.html',ans=pred)

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



#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.y_predict ( [np.array(list(data.values()))])

    output = prediction[0]
    return jsonify (output)

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

7.3 Database Schema (if Applicable)





CHAPTER-8

8.TESTING

8.1 Test Cases

				Date	15-Nov-22								
				Team ID	PNT2022TMID32563								
				Project Name	Project-WebPhishing Detection								
				Maximum Marks	4marks								
Test case ID	Feature Type	Component	TestScenario	Pre-Requisite	Steps To Execute	TestData	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go 2.Type the URL 3.Verify whether it is pr ocessing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Workingas expected	Pass		N		Subasri R
LoginPage_TC_OO2	UI	Home Page	Verify the UIeleme nts is Responsive		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check whether the button is responsive or not 4. Reload and TestSimultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then getsAcknowledge	Workingas expected	Pass		N		Swetha A
LoginPage_TC_OO3	Functional	Home page	Verify whether the link is legitimate ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the websiteislegitimate ornot.	Workingas expected	Pass		N		Swetha D
LoginPage_TC_OO4	Functional	Home Page	Verify user is able to access the legitimate website ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if itis notlegitimate.	https://phishing-shield.herokuapp.com/	Application shouldshow that Safe Webpage orUnsafe.	Workingas expected	Pass		N		Sadhana M
LoginPage_TC_OO5	Functional	Home Page	Testing the website with multiple URLs		1. Enter URL (https://phishing-shield.herokuapp.com /) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure orbe cautiousif itis not secure	1. https://aybalajee.github.io/welcome 2.totalpad.com https://www.kincp.edu salescript.info 5. https://www.google.com/ 6.delgets.com	User can able to identify the websites whether it is secure or not	Workingas expected	Pass		N		Subasri R Swetha A Swetha D Sadhana M

8.2 User Acceptance Testing

UAT Execution & Report Submission

1. **Purpose of Document**

The purpose of this document is to briefly explain the test coverage and open issues of the [Web Phishing Detection] project at the time of the release to User Acceptance Testing (UAT).

2. **Defect Analysis**

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

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	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	60

3.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	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

CHAPTER-9

9.RESULTS

9.1 Performance Metrics

S.No	Parameter	Values
1.	Metrics	**Classification Model:** **Random Forest Classifier** Accuracy Score-95%
2.	Tune the model	Hyperparameter Tuning-97% Validation Method-KFOLD& Cross Validation Method

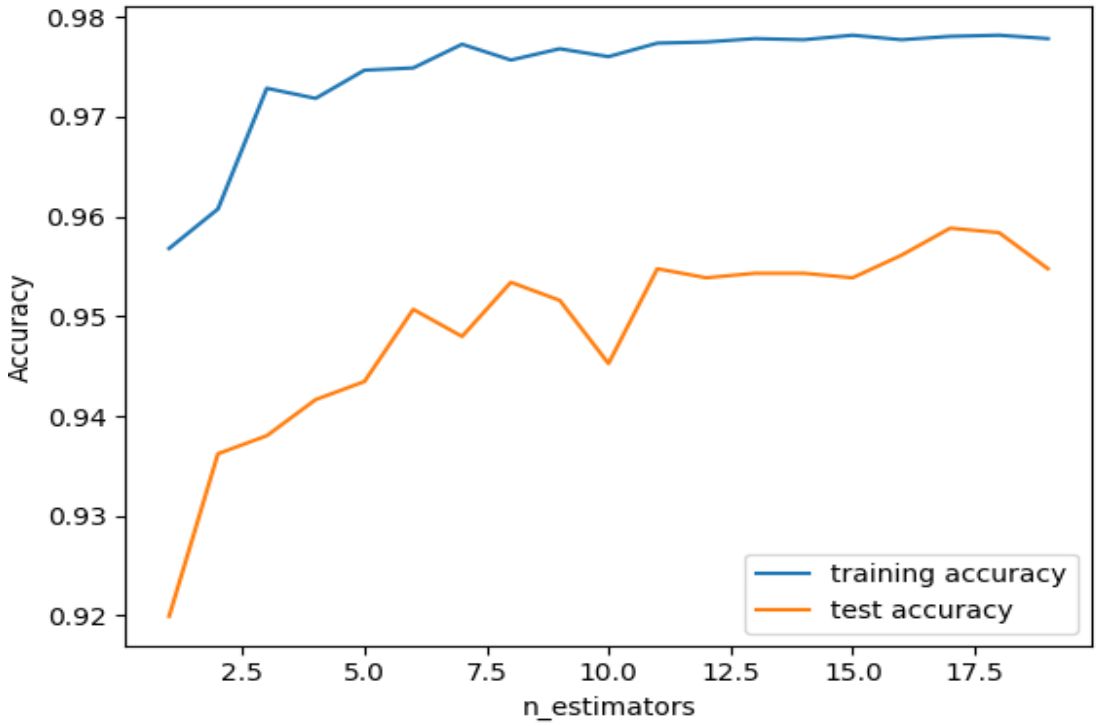
1.METRICS:

CLASSIFICATION REPORT:

```
In [54]: #computing the classification report of the model

         print(metrics.classification_report(y_test, y_test_forest))
```

```
              precision    recall  f1-score   support

          -1       0.96      0.99      0.97      1933
           1       0.87      0.70      0.78       278

    accuracy                           0.95      2211
   macro avg       0.92      0.84      0.88      2211
weighted avg       0.95      0.95      0.95      2211
```

PERFORMANCE:



Out[92]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	0.884	0.264	0.133	0.585
1	K-Nearest Neighbors	0.943	0.778	0.902	0.898
2	Support Vector Machine	0.916	0.526	0.457	0.964
3	Naive Bayes Classifier	0.559	0.362	0.993	0.242
4	Decision Tree	0.949	0.785	0.869	0.968
5	Random Forest	0.950	0.778	0.864	0.956
6	Gradient Boosting Classifier	0.945	0.750	0.756	0.921
7	CatBoost Classifier	0.960	0.827	0.875	0.948
8	XGBoost Classifier	0.087	0.087	0.782	0.924
9	Multi-layer Perceptron	0.085	0.085	0.766	0.917
10	Multi-layer Perceptron	0.085	0.085	0.766	0.917

2. TUNE THE MODEL – HYPERPARAMETER TUNING

In [58]:
```
#HYPERPARAMETER TUNING
grid.fit(X_train, y_train)
```

Out[58]:
```
GridSearchCV

GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                  max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})

estimator: GradientBoostingClassifier

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

GradientBoostingClassifier

GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

In [59]:
```
print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))
```

```
The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97
```

VALIDATION METHODS: KFOLD & Cross Folding

Wilcoxon signed-rank test

In [78]:
```
#KFOLD and Cross Validation Model

from scipy.stats import wilcoxon
from sklearn.datasets import load_iris
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score, KFold

# Load the dataset
X = load_iris().data
y = load_iris().target

# Prepare models and select your CV method
model1 = GradientBoostingClassifier(n_estimators=100)
model2 = XGBClassifier(n_estimators=100)
kf = KFold(n_splits=20, random_state=None)
# Extract results for each model on the same folds
results_model1 = cross_val_score(model1, X, y, cv=kf)
results_model2 = cross_val_score(model2, X, y, cv=kf)
stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
stat
```

Out[78]: 95.0

5x2CV combined F test

In [89]:
```
from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                            estimator2=clf2,
                            X=X, y=y,
                            random_seed=1)

print('f-value:', f)
print('p-value:', p)
```

```
f-value: 1.727272727272733
p-value: 0.2840135734291782
```

CHAPTER -10

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 goodcustomer relationships.
6. If internet connection fails this system will work

Disadvantages of web phishingdetection

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

CHAPTER 11

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 tacklea wide assortment of classification issues, the procedure of finding the ideal structure is verydifficult, 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-12

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-13

Appendix:

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

Source Code:

Homepage.html:

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>WEB PHISHING DETECTION</title>



    <style>
        *{
    margin: 0;
    padding: 0;
    font-family: Century Gothic;
}
body{
    background: linear-gradient(rgba(0, 0, 50, 0.5),rgba(0, 0, 50, 0.5)),url(https://media.istockphoto.com/id/1335959802/photo/ransomware-cyber-security-email-phishing-encrypted-technology-digital-information-protected.jpg?s=612x612&w=0&k=20&c=9LgCSouqRqbAeJzDXTkRE8O6T74eJwTmGMKBxiOSS5E=);
    background-size: 1400px 650px;

}
ul{
    float: right;
    list-style-type: none;
```

```
    margin: 25px;
}
ul li{
    display: inline-block;
}

ul li a{
    text-decoration: none;
    color: #fff;
    padding: 5px 20px;
    border: 1px solid transparent;
    transition: 0.6s ease;
}

ul li a:hover{
    background-color: #fff;
    color: #000;
}
ul li.active a{
    background-color: #fff;
    color: #000;
}
.main{
    max-width: 1200px;
    margin: auto;
}
.title{
    position: absolute;
    top: 50%;
    left: 50%;
```

```
    transform: translate(-50%,-50%);
}
.title h1{
    color: #fff;
    font-size: 40px;
}
    </style>

</head>
<body>

        <div class="main">
            <div class="links">
                <ul>
                <li class="active"><a href="#" >Home</a></li>
                <li><a href="{{ url_for('register') }}">REGISTER</a></li>
                <li><a href="{{ url_for('login') }}">LOGIN</a></li>
                <li><a href="{{ url_for('about') }}">ABOUT</a></li>

            </ul>
                        </div>

        </div>
        <div class="title">
            <h1>WEB PHISHING DETECTION</h1>
        </div>
</body>
</html>
```

About.html:

```
<html>
   <head>
      <h3>ABOUT</h3>
   </head>
   <style>
      .container {
   width: 100%;
   padding: 20px 0;
  }

  .container h2 {
   margin-top: 4rem;
   font-size: 42px;
   text-align: center;
   color: rgb(5, 141, 182);
   text-transform: capitalize;
   text-overflow: hidden;
  }

  .text-area {
   height: 200px;
   width: 100%;
   max-width: 100%;
   display: flex;
   align-items: center;
   justify-content: space-around;
  }
```

```
  .text1,
  .text2 {
   padding: 10px 30px;
   letter-spacing: 0.5px;
   font-size: 17px;
   color: white;
  }
  body{
   background: linear-gradient(rgba(0, 0, 50, 0.5),rgba(0, 0, 50, 0.5)),url(https://bangitsolutions.com/wp-content/uploads/2020/11/phishing-credit-card-data-1536x1024.jpg);
   background-size: 1400px 650px;

}

  @media (min-width: 769px) {

   .header,
   .main-nav {
   display: flex;
   }

   .header {
   flex-direction: column;
   align-items: center;
   }

  }
```

```
   @media (min-width: 1025px) {
    .header {
    flex-direction: row;
    justify-content: space-between;
    }
    }
    p a{

    list-style-type: none;
    margin: 25px;
    position: absolute;
    top: 5%;
    left: 85%;
}
p a{
    display: inline-block;
    background-color: rgb(243, 250, 144);
}

p a{
    text-decoration: none;
    color:  rgb(0, 78, 74);
    padding: 5px 20px;
    border: 1px whitesmoke;
    transition: 0.6s ease;
    font-weight: 700;
}
    </style>
```

```
<body>
    <div class="container" id="about">
        <h2 class="title" id="h2">ABOUT</h2>
        <div class="text-area">
            <div class="text1">
                <p>Phishing attacks are one of the most common form of social engineering attacks. In a web-based
                    phishing attack, attackers use web pages visually mimicking legitimate web sites, such as banking
                    and government services, to deceive the victims to input their sensitive information (e.g., bank
                    accounts and social security number). Though phishing attacks do not require advanced technical
                    knowledge and these attack techniques are becoming familiar to users, they are still causing major
                    financial damages.</p>
            </div>
            <div class="text2">
                <p>In several fields, automation has been achieved through the extensive use of machine learning. Our approach is based on the aggregate analysis method to automatically develop rules to determine layout similarity of web sites and then detect phishing pages. Researchers also utilise machine learning to detect phishing assaults based on numerous aspects. Our strategy is divided into two parts. It leverages the attributes of the website layout to first train a similarity classifier, which is then used to identify phishing pages..
                </p>
            </div>
        </div>

        <p><a href="{{ url_for('homepage') }}">GO BACK</a></p>
```

```
        </div>
    </body>
</html>
```

Reg.html:

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Registeration Form</title>
<link                                                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
</head>
<style>
        body{
            background: linear-gradient(rgba(0, 0, 50, 0.5),rgba(0, 0, 50,
0.5)),url(https://imageio.forbes.com/specials-
images/imageserve/5f7cdc22235350fb9f68821a/0x0.jpg?format=jpg&width=120
0);
            background-size: 1400px 650px;
            font-family: Arial,Arial, Helvetica, sans-serif;
            color: white;
       }

</style>
<body>
<div class="container">
        <h2>User Registration</h2>
        <form action="{{ url_for('register') }}" method="post">
```

```
    {% if msg is defined and msg %}
                <div class="alert alert-warning">{{ msg }}</div>
            {% endif %}
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" class="form-control" id="name" name="name" placeholder="Enter name" name="name">
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" class="form-control" id="email" name="email" placeholder="Enter email" name="email">
            </div>
            <div class="form-group">
                <label for="pwd">Password:</label>
                <input type="password" class="form-control" id="password" name="password" placeholder="Enter password" name="pswd">
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
            <p class="bottom">Already have an account?  <a class="bottom" href="{{url_for('login')}}"> Login here</a></p>


        </form>
</div>
</body>
```

Login.html:

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>User Login Form</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
</head>
<style>
        body{
            background: linear-gradient(rgba(0, 0, 50, 0.5),rgba(0, 0, 50, 0.5)),url(https://t3.ftcdn.net/jpg/00/54/46/70/360_F_54467041_vwkt3lzq2365E05fI7YRxBxCI1aVxcTj.jpg);
            background-size: 1400px 650px;
            font-family: Arial,Arial, Helvetica, sans-serif;
            color: white;
      }

</style>
<body>
<div class="container">
        <h2>User Login</h2>
        <form action="{{ url_for('login') }}" method="post">
           {% if msg is defined and msg %}
                    <div class="alert alert-warning">{{ msg }}</div>
              {% endif %}
              <div class="form-group">
                    <label for="email">Email:</label>
```

```
                    <input type="email" class="form-control" id="email" name="email" placeholder="Enter email" name="email">
                </div>
                <div class="form-group">
                    <label for="pwd">Password:</label>
                    <input type="password" class="form-control" id="password" name="password" placeholder="Enter password" name="pswd">
                </div>
                <button type="submit" class="btn btn-primary">Login</button>
                <p class="bottom">Dont't have an account?  <a class="bottom" href="{{url_for('register')}}"> Register here</a></p>
            </form>
</div>
</body>
</html>
```

Predictform.html:

```
<html>
   <style>
      *{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: Century Gothic;
}
body{
   background: linear-gradient(rgba(0, 0, 50, 0.5),rgba(0, 0, 50, 0.5)),url(https://static.vecteezy.com/system/resources/previews/002/202/047/original/blue-high-tech-futuristic-cyberspace-technology-background-free-vector.jpg);
```

```
    background-size: 1400px 650px;
}
.title h8{
    color: #fff;
    font-size: 20px;
    top: 30%;
    left: 17%;
    position: absolute;
}
.title h5{
    color: #fff;
    font-size: 40px;
    top: 20%;
    left: 37%;
    transform: translate(-20%,-37%);
    position: absolute;
}

#pbutton{
            width: 300px;
            height: 30px;
            border: none;
            background-color: rgba(17, 139, 238, 0.747);
            border-radius: 10px;
            padding-top: 5px;
            color: whitesmoke;
            font-size: 16px;
            position: absolute;
            top:60%;
            left:37%;
```

```
            padding-bottom: 5px;

            }

input[type="text"]{
            width: 50%;
            height: 3px;
            padding: 20px;
            margin: 5px 0 22px 0;
            display: inline-block;
            outline: black;
            outline-style: auto;
            background-color: whitesmoke;
            align-self: auto;


}
.search-box{
    display: flex;
    align-items: center;
    justify-content: center;
    margin: auto;
    width: 100%;
    height: 100vh;

}
ul{
    float: right;
    list-style-type: none;
    margin: 25px;
```

```
}

ul li{

    display: inline-block;
}

ul li a{
    text-decoration: none;
    color: rgb(8, 70, 46);
    padding: 5px 20px;
    border: 1px solid transparent;
    transition: 0.6s ease;
}
p a{

    list-style-type: none;
    margin: 25px;
    position: absolute;
    top: 5%;
    left: 85%;
}
p a{
    display: inline-block;
    background-color: rgb(243, 250, 144);
}

p a{
    text-decoration: none;
    color: rgb(0, 78, 74);
```

```
    padding: 5px 20px;
    border: 1px whitesmoke;

    transition: 0.6s ease;
    font-weight: 700;

}
    </style>

    <body>

        <section class="about">
            <div class="main">
                <div class="title">

                    <h5>Phishing Website Detection</h5>

                    <h8>Beware of phishing websites that are taking your sensitive information and login passwords.
                    </h8>
                    <form action="/y_predict" method="POST">
                    <div class="search-box">
                        <input class="search-txt" type="text" name="url" id="url" placeholder="Enter your link">
                        <i class="fas fa-search"></i>
                    </div>
                    <input type="submit" name="PREDICT" id="pbutton" value="PREDICT" >
                </form>
                </div>
```

```
            </div>
        </section>
    <h3 style="text-align:center; color:rgb(4, 207, 4); font-size:20px; position: absolute; top: 70%;left:35%">{{ans}}</h3>


    <h3 style="text-align:center; color:rgb(216, 0, 0); font-size:20px; position: absolute; top: 70%;left:35%">{{bns}}</h3>

        <p><a href="{{ url_for('logout') }}">LOGOUT</a></p>

</body>
</html>
```

App.py:

```
import pickle
from flask import Flask, jsonify, render_template, request,redirect,url_for,request

#importing the inputScript file used to analyze the URL import inputScript
import inputScript
import numpy as np
import sklearn

from flask import Flask, render_template, request, redirect, url_for, session

from flask_mysqldb import MySQL

import MySQLdb.cursors
```

```
import re
app = Flask(__name__)

app.secret_key = 'web'

app.config['MYSQL_HOST'] = 'localhost'

app.config['MYSQL_USER'] = 'root'

app.config['MYSQL_PASSWORD'] = 'Suba@10'

app.config['MYSQL_DB'] = 'webdb'

app.config['MYSQL_CURSORCLASS'] = 'DictCursor'


mysql = MySQL(app)

#load model

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

@app.route('/')
@app.route('/homepage')
def homepage():
    return render_template('homepage.html')


@app.route('/flask/login', methods =['GET', 'POST'])
def login():
```

```
    msg = ''
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        email = request.form['email']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM signup WHERE email = % s AND password = % s', (email, password, ))
        user = cursor.fetchone()
        if user:
            session['loggedin'] = True
            session['name'] = user['name']
            session['email'] = user['email']
            mesage = 'Logged in successfully !'
            return render_template('predictform.html', msg = msg)
        else:
            mesage = 'Please enter correct email / password !'
    return render_template('login.html', msg = msg)


@app.route('/flask/reg', methods =['GET', 'POST'])

def register():
    msg = ''

    if request.method == 'POST' and 'name' in request.form  and 'email' in request.form and 'password' in request.form:

        name = request.form['name']
```

```
        email = request.form['email']

        password = request.form['password']

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT * FROM signup WHERE email = % s', (email, ))

        account = cursor.fetchone()

        if account:

            msg = 'Account already exists !'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

            msg = 'Invalid email address !'

        elif not re.match(r'[A-Za-z0-9]+', name):

            msg = 'Username must contain only characters and numbers !'

        elif not name or not password or not email:

            msg = 'Please fill out the form!'

        else:

            cursor.execute('INSERT INTO signup
VALUES(%s,%s,%s)',(name,email,password,))
```

```
            mysql.connection.commit()

            msg = 'You have successfully registered !'

            return redirect(url_for('login'))

    elif request.method =='POST':
            msg='Please fill out the form!'

    return render_template('reg.html', msg = msg)

@app.route('/flask/predictform', methods =['GET', 'POST'])
def predictform():
    return render_template('predictform.html')

@app.route('/flask/about', methods =['GET', 'POST'])
def about():
    return render_template('about.html')

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('name', None)
    session.pop('email', None)
    return redirect(url_for('homepage'))

#Redirects to the page to give the user iput URL.
@app.route('/flask/result')
def result():
    return render_template('result.html')
```

```
ans=""
bns=""

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict', methods=['POST'])
def y_predict():

    '''
    For rendering results on HTML GUI
    '''

    url = request.form['url']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction [0]
    if(output==1):
        pred="Your are safe!! This is a Legitimate Website."
        return render_template('result.html',ans=pred)

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



#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.y_predict ( [np.array(list(data.values()))])

    output = prediction[0]
    return jsonify (output)

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

InputScript.py:

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

```
"""
Check 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
"""
Check for the URL length. Return 1 (Legitimate) if the URL length is less than 54
characters
Return 0 if the length is between 54 and 75
Else return -1
"""
def URLURL_Length (url):
   length=len(url)
   if(length<=75):
       if(length<54):
          return 1
       else:
          return 0
```

```
    else:
        return -1

"""
Check with the shortened URLs.
Return -1 if any shortened URLs used.
Else return 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\.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',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

"""
Check the Subdomain. Return 1 if the subDomain contains less than 1 '.'
```

```
Return 0 if the subDomain contains less than 2 '.'
Return -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 and 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(), "lxml").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\.40|83\.125\.22\.219|46\.242\.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.1
```

```
96\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198
\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|2
09\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|5
4\.86\.225\.156|54\.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),SSLfin
al_State(url),

Domain_registeration_length(url),Favicon(url),port(url),HTTPS_token(url),Requ
est_URL(url),

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abn
ormal_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_In
dex(url),
        Links_pointing_to_page(url),Statistical_report(url)]]


    print(check)
    return check
```

Project Link: https://github.com/IBM-EPBL/IBM-Project-30500-1660147496

Project Demo Link:
https://drive.google.com/file/d/146aOIKvQfjIdyZYZ0g3BxQ-1oBnQk6By/view?usp=drivesdk