WEB PHISHING DETECTION

A PROJECT REPORT

Submitted by

SAMPAHKUMAR P

SANJAIKUMAR S

DHARINEESH B

JANAN C

of

COMPUTER SCIENCE AND ENGINEERING

Dr. MAHALINGAM COLLEGE OF ENGINEERING AND TECHNOLOGY

POLLACHI – 642003

CHAPTER 1

INTRODUCTION

1.1 Project Overview:

This project mainly focuses on building a machine learning model to detect the phishing websites. To do this we have developed an effective model using classification algorithm . 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:

Phishing is a form of fraudulent attack where the attacker tries to gain sensitive information by posing as a reputable source. In a typical phishing attack, a victim opens a compromised link that poses as a credible website. The victim is then asked to enter their credentials, but since it is a "fake" website, the sensitive information is routed to the hacker and the victim gets "'hacked."

Phishing is popular since it is a low effort, high reward attack also affects may person personal life and this model will be helpful in identifying the type of URL entered by the user

.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem:

The purpose or goal behind phishing is data, money or personal information stealing through the fake website. The best strategy for avoiding the contact with the phishing web site is to detect real time malicious URL. Phishing websites can be determined on the basis of their domains.

They usually are related to URL which needs to be registered (low-level domain and upper-level domain, path, query). Recently acquired status of intra-URL relationship is used to evaluate it using distinctive properties extracted from words that compose a URL based on query data from various search engines such as Google and Yahoo. These properties are further led to the machine-learningbased classification for the identification of phishing URLs from a real dataset.

This paper focus on real time URL phishing against phishing content by using phish-STORM. For this a few relationship between the register domain rest of the URL are consider also intra URL relentless is consider which help to dusting wish between phishing or non phishing URL.

For detecting a phishing website certain typical blacklisted urls are used, but this technique is unproductive as the duration of phishing websites is very short. Phishing is the name of avenue. It can be defined as the manner of deception of an organization's customer to communicate with their confidential information in an unacceptable behaviour. It can also be defined as intentionally using harsh weapons such as Spasm to automatically target the victims and targeting their private information.

As many of the failures being occurred in the SMTP are exploiting vectors for the phishing websites, there is a greater availability of communication for malicious message deliveries. Proposed a novel classification approach that use heuristic based feature extraction approach. In this, they have classified extracted features into different categories such as URL Obfuscation features, Hyperlink-based features. Moreover, proposed technique gives 92.5% accuracy. Also thismodel is purely depends on the quality and quantity of the training set and Broken links feature extraction

2.2 References:

[1]. Gunter Ollmann, "The Phishing Guide Understanding & Preventing Phishing Attacks", IBMInternet Security Systems, 2007.

[2]. https://resources.infosecinstitute.com/category/enterprise /phishing/the-phishing-landscape/phishing-data-attackstatistics/#gref

[3]. Mahmoud Khonji, Youssef Iraqi, "Phishing Detection: A Literature Survey IEEE, and Andrew Jones, 2013

[4]. Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites Accessed January 2016

[5]. http://dataaspirant.com/2017/01/30/how-decision-treealgorithm-works/

[6]. http://dataaspirant.com/2017/05/22/random-forestalgorithm-machine-learing/

[7]. https://www.kdnuggets.com/2016/07/support-vectormachines-simple-explanation.html

[8]. www.alexa.com

[9]. www.phishtank.com

[10]. Anti-phishing Working Group (APWG) Phishing Activity Trends Report 4th quarter 2020, https://docs.apwg.org/reports/apwg trends report q4 2020.pdf

2.3 Problem statement definition:

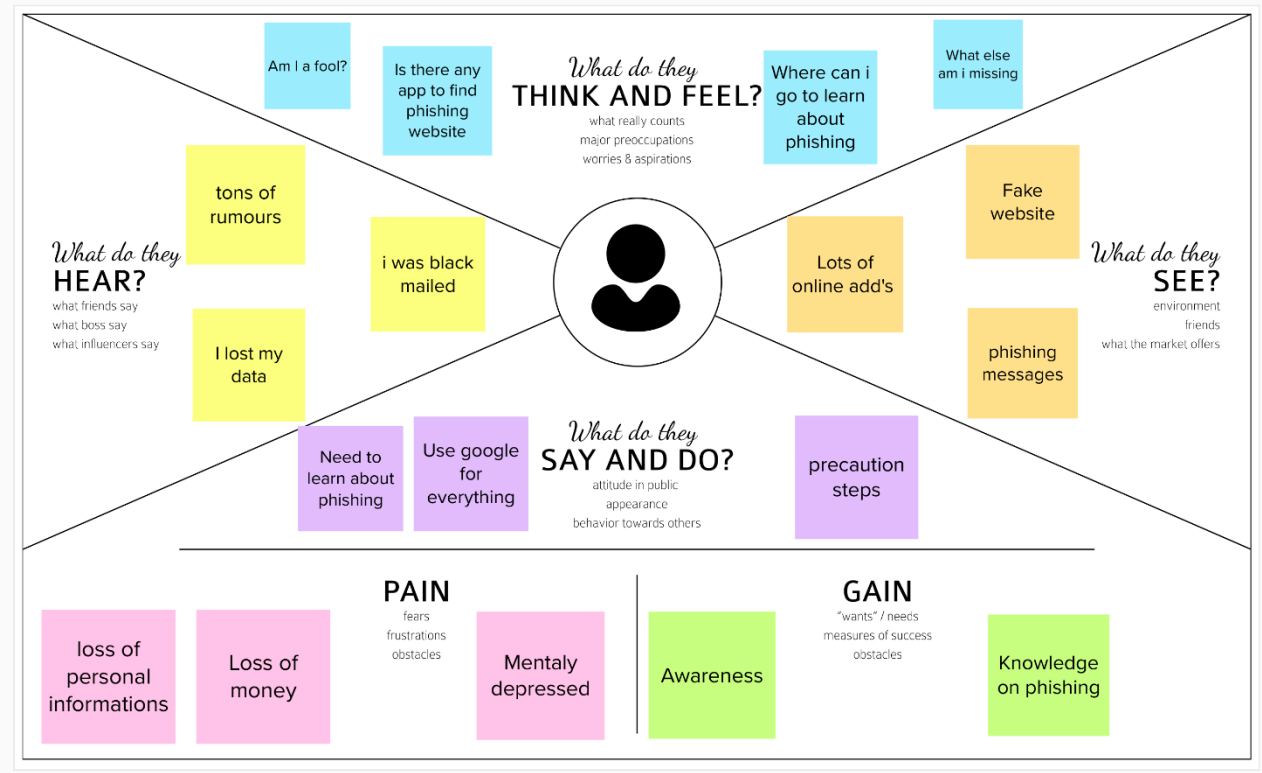
Phishing is one of the techniques which are used by the intruders to get access to the user credentials or to gain access to the sensitive data. This type of accessing the is done by creating the replica of the websites which looks same as the original websites which we use on our daily basis but when a user click on the link he will see the website and think its original and try to provide his credentials . To overcome this problem we are using some of the machine learning algorithms in which it will help us to identify the phishing websites based on the features present in the algorithm. By using these algorithm we cam be able to keep the user personal credentials or the sensitive data safe from the intruders.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

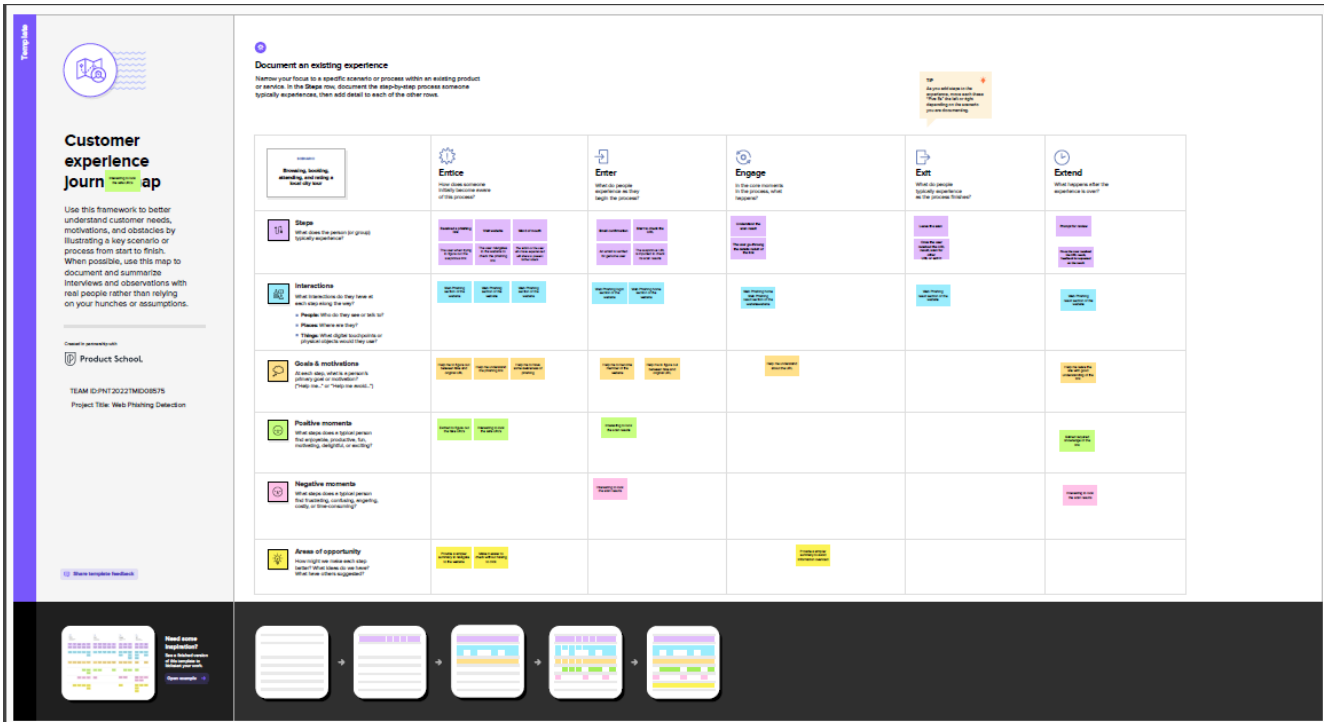
An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. Empathy maps should be used throughout any UX process to establish common ground among team members and to understand and prioritize user needs. In user-centered design, empathy maps are best used from the very beginning of the design process.



3.2 Ideation & Brainstorming:

Ideation essentially refers to the whole creative process of coming up with and communicating new ideas. Ideation is innovative thinking, typically aimed at solving a problem or providing a more efficient means of doing or accomplishing something.

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



3.3 **Proposed Solution:**

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Phishing is the most popular attack vector for criminals and has grown 65% in the last year, we could not stop this completely but we can avoid most of them by the use of our proposed system that use machine learning
2.	Idea / Solution description	The phishing website can be detected based on some important characteristics like link and Domain names, and security and encryption criteria in the final phishing detection rate.

3.	Novelty / Uniqueness	There are many technical solutions to protect ourself from this phishing attack, But the date mining algorithm used in this system provides comparatively better performance as compared to other traditional classification algorithms
4.	Social Impact / Customer Satisfaction	This proposed system helps the customer to make online transactions without fear and a confidence of how to avoid phishing, and will be aware of what is phishing and how their data is getting stolen.
5.	Business Model (Revenue Model)	The no of users of this website will increase the promotion of this system, and will lead to promote it as a brand .
6.	Scalability of the Solution	In fracture the progress can be able to detect phishing attack based on the IP address and file attachments and so on …,

6.	Scalability of the Solution	This application can be accessed online without paying. It can be accessed via any browser of your choice. It can detect any site with high accuracy.

3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioural patterns and recognize what would work and why.

Purpose:

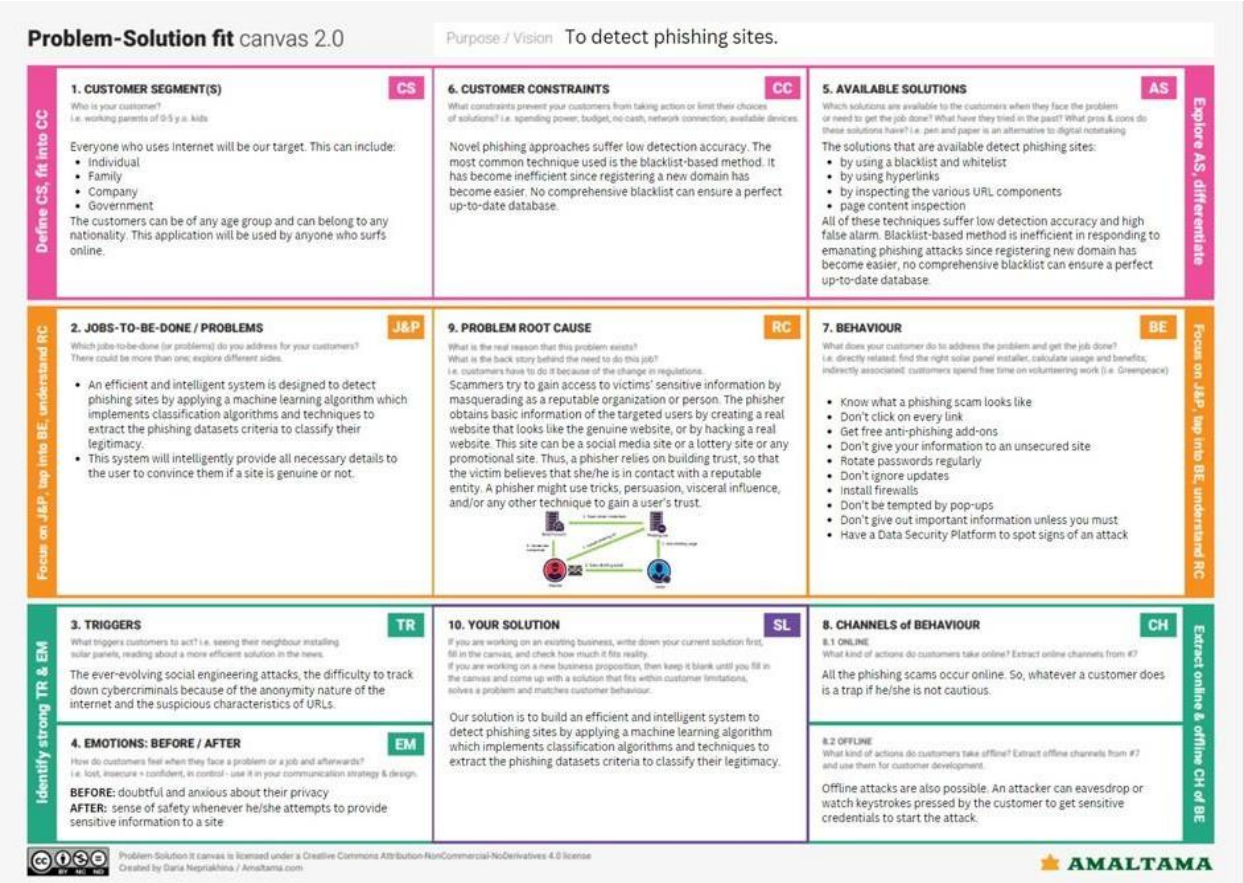
❑ Solve complex problems in a way that fits the state of your customers.

❑ Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behaviour.

❑ Sharpen your communication and marketing strategy with the right triggers and messaging.

❑ Increase touchpoints with your company by finding the right problem-behaviour fit and building trust by solving frequent annoyances, or urgent or costly problems.

❑ Understand the existing situation in order to improve it for your target group.



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirements:

FR No.	Functional Requirement (Epic)	Description
FR-1	User Input	User inputs an URL in the form to check whether it is a malicious website.
FR-2	Website comparison	The model compares the given URL with the list of phishing URLs present in the database.
FR-3	Feature Extraction	If it is found none on the comparison it extracts the HTML and domain-based features from the URL.
FR-4	Prediction	The model predicts the URL using machine Learning algorithms such as Random Forest technique.
FR-5	Classifier	Model then sends the output to the classifier and produces the result.
FR-6	Announcement	The model finally displays whether the given URL is phishing or not.

4.2 Non-functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It is an easy to use and access interface which results in greater efficiency.
NFR-2	Security	It is a secure website which protects the sensitive information of the user and prevents malicious attacks.
NFR-3	Reliability	The system can detect phishing websites with greater accuracy using ML algorithms.
NFR-4	Performance	The system produces responses within seconds and execution is faster.
NFR-5	Availability	Users can access the website via any browser from anywhere at any time.
NFR-6	Scalability	This application can be accessed online without paying. It can detect any web site with high accuracy.

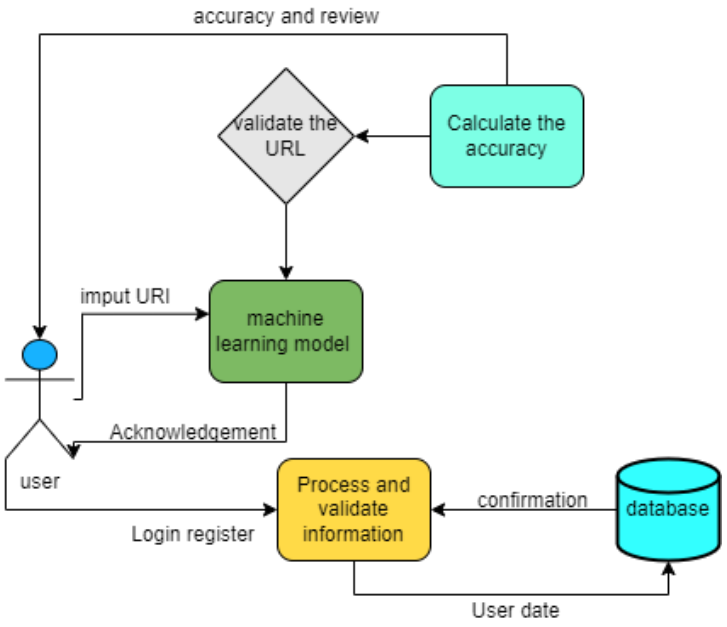
CHAPTER 5

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.

DFD level 0:



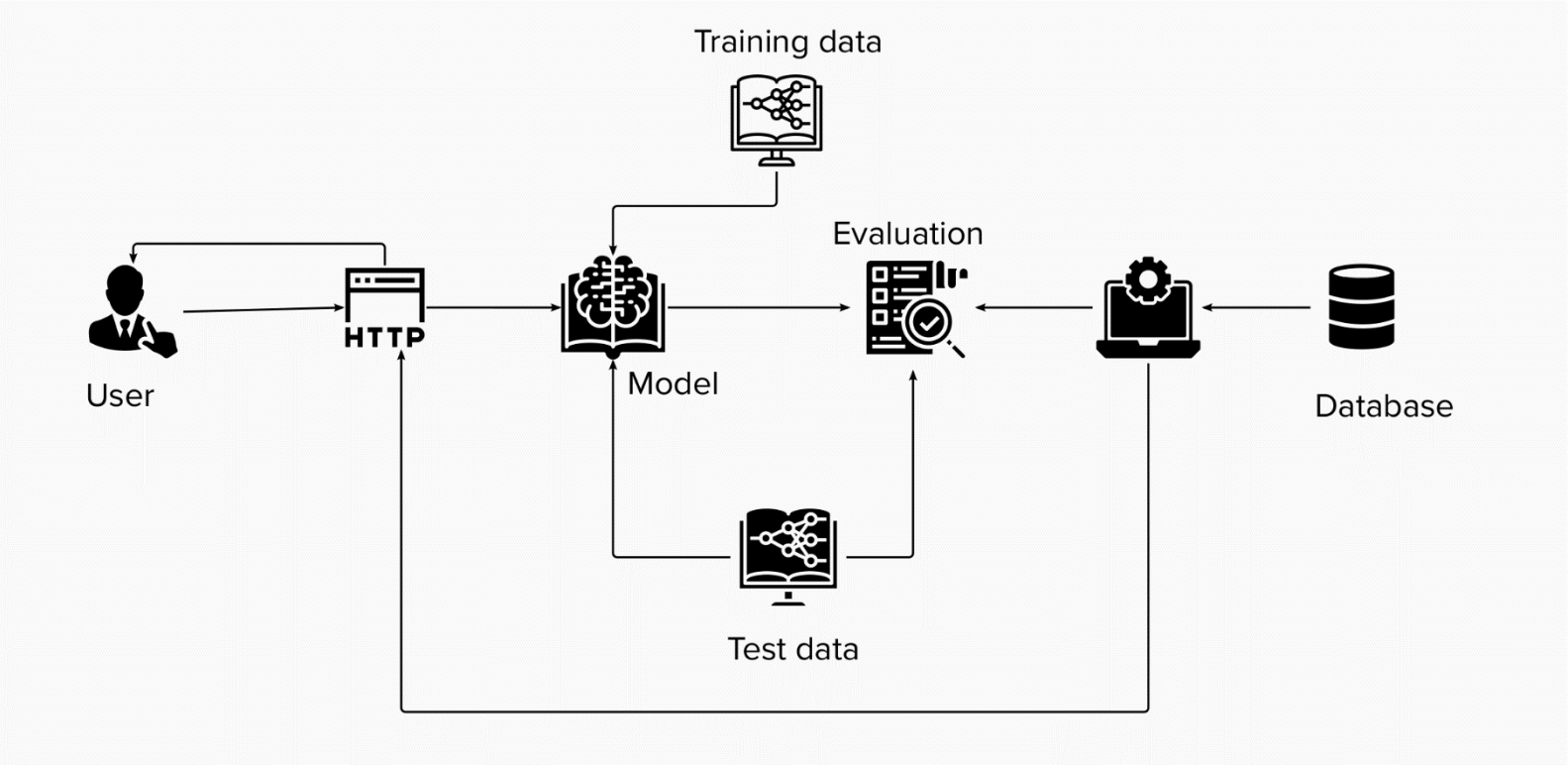
5.2 Solution & Technical Architecture:

SOLUTION:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Solution Architecture Diagram：



5.3 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my 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 Gmail	I can register and create my profile	Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access the website features	High	Sprint-2
	User input	USN-5	User can type or paste the URL in the given input box		High	Sprint-3
Administrator	Data collection	USN-6	Various URL data were collected, both secure and in-secure	Data can be cleaned and used	High	Sprint-3
	Data pre-processing	USN-7	Unwanted data is to be removed, and cleaned so that the result will be faster and accurate	Data can be used to train the model without any further transformation	High	Sprint-4
	Model building	USN-8	A machine learning model is built with the available dataset	Model is ready to deployment	High	Sprint-5
	Application building	USN-9	The user need not to code anything as every actions will be designed with familiar User interface	Easy to understand the process, and easy to work with User Interface	High	Sprint-5

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Number Story	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Collection of dataset	USN-1	As a developer ,I need to collect related data and to store it in digital format to make machine learning models to understand	3	High	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.
Sprint-1	Data Pre-processing	USN-2	As a developer , I need to clean and organize the raw data to make it suitable for building and training the model	8	High	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.
Sprint-2	Exploratory Data Analysis	USN-3	As a developer ,EDA approach is used to analyze the data to shortlist the relevant columns required to train the model	5	Medium	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.
Sprint-3	Model building	USN-4	As a developer ,I need to use the dataset to built a model and test it using the test dataset	13	Hifh	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.
Sprint-4	UI Designing	USN-5	As a developer ,I need to provide the user a good experiencing user interface	3	Medium	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.
Sprint-4	UI Integration	USN-6	As a developer , I need to integrate UI page and the model to get user input and display the result in more user-friendly manner.	8	High	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.

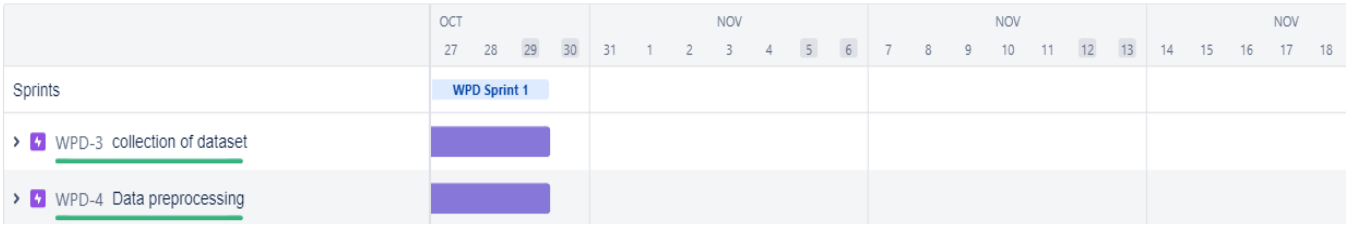
Sprint 4	Result	USN-7	User can visit the website and enter the URL to be suspected and the result will be displayed (whether trustworthy or not)	5	Medium	Sampathkumar P, Sanjaikumar S. Janan C, Dharineesh B.

6.2 Sprint Delivery Schedule:

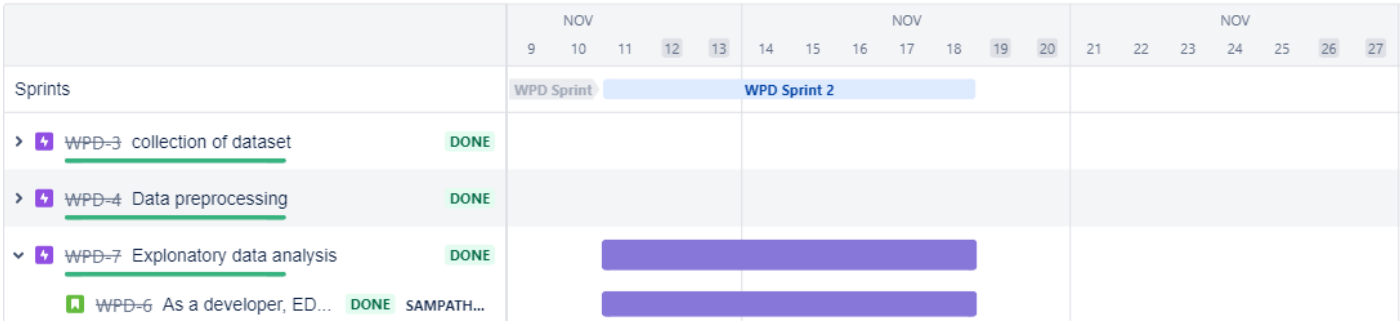
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	11	6 Days	24 Oct 2022	29 Oct 2022	11	31 Oct 2022
Sprint-2	5	6 Days	31 Oct 2022	05 Nov 2022	5	14 Nov 2022
Sprint-3	13	6 Days	07 Nov 2022	12 Nov 2022	13	17 Nov 2022
Sprint-4	16	6 Days	14 Nov 2022	19 Nov 2022	11	18 Nov 2022

6.3 Reports from JIRA:

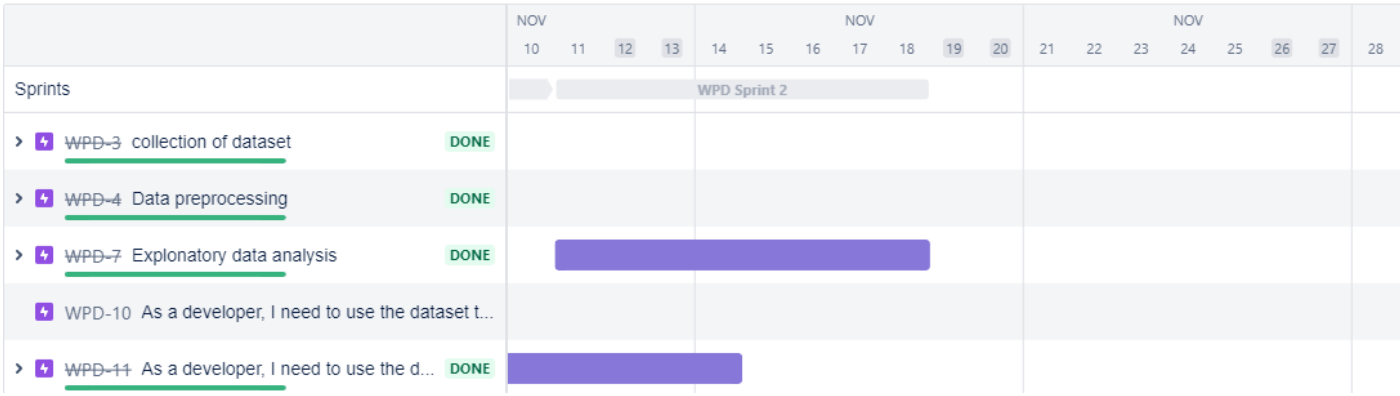
Sprint 1:



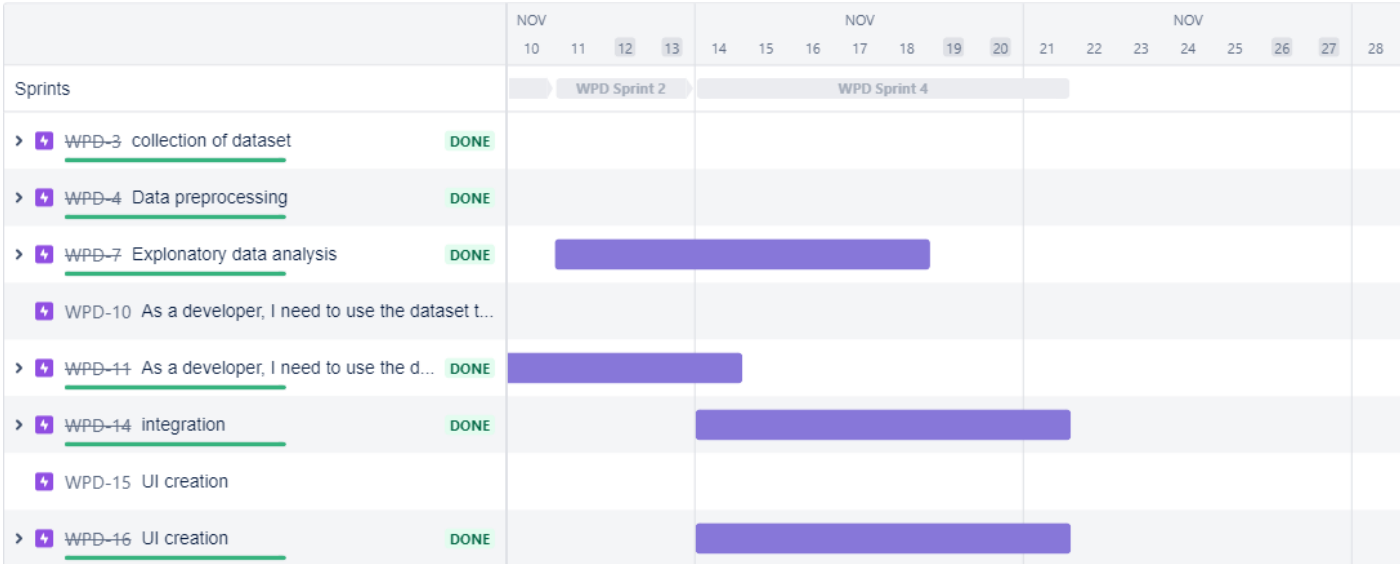
Sprint 2:



Sprint 3:



Sprint 4:



CHAPTER 7

CODING & SOLUTIONING

7.1 Feature 1

Data collection: Feature extraction

URL features of legitimate websites and phishing websites were collected. The data set consists of total 11,055 URLs which include 6,157 legitimate URLs and 4,898 phishing URLs. Legitimate URLs are labelled as “1” and phishing URLs are labelled as “-1”. The features that are present in the data set include:

- IP Address in URL
- Length of URL
- Using URL Shortening Services
- "@" Symbol in URL
- Redirection "//" in URL
- Prefix or Suffix "-" in Domain
- Having Sub Domain
- Length of Domain Registration
- Favicon
- Port Number
- HTTPS Token
- Request URL
- URL of Anchor
- Links in Tags
- SFH
- Email Submission
- Abnormal URL
- Status Bar Customization (on mouse over)
- Disabling Right Click
- Presence of Popup Window
- IFrame Redirection
- Age of Domain
- DNS Record
- Web Traffic
- Page Rank
- Google Index
- Links pointing to the page
- Statistical Report
- Result

Using IBM Cloud Storage this data is accessed throughout the project. The code written below is used to import the dataset.

Data Preprocessing:

Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance

In [20]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score as accuracy
import pickle
```

In [2]:
```
data = pd.read_csv("dataset_website.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSl
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 32 columns

In [8]:
```
i=data.drop(["index"],axis=1)
```

In [9]:
```
i.head()
```

Out[9]:

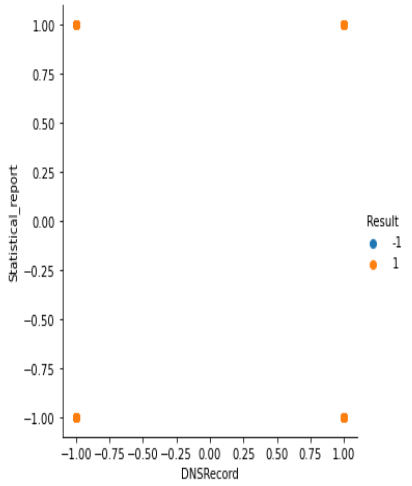
	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_S
0	-1	1	1	1	-1	-1	-1	
1	1	1	1	1	1	-1	0	
2	1	0	1	1	1	-1	-1	
3	1	0	1	1	1	-1	-1	
4	1	0	-1	1	1	-1	1	

5 rows × 31 columns

Find out the dependent variables:

We compare any of the independent variable with the dependent variable and to find out the rate of dependency

```
In [11]: import matplotlib.pyplot as plt
         import seaborn as seb

In [12]: seb.FacetGrid(data, hue = 'Result', height=5).map(plt.scatter,'DNSRecord','Statistical_report').add_legend()
         plt.show()
```

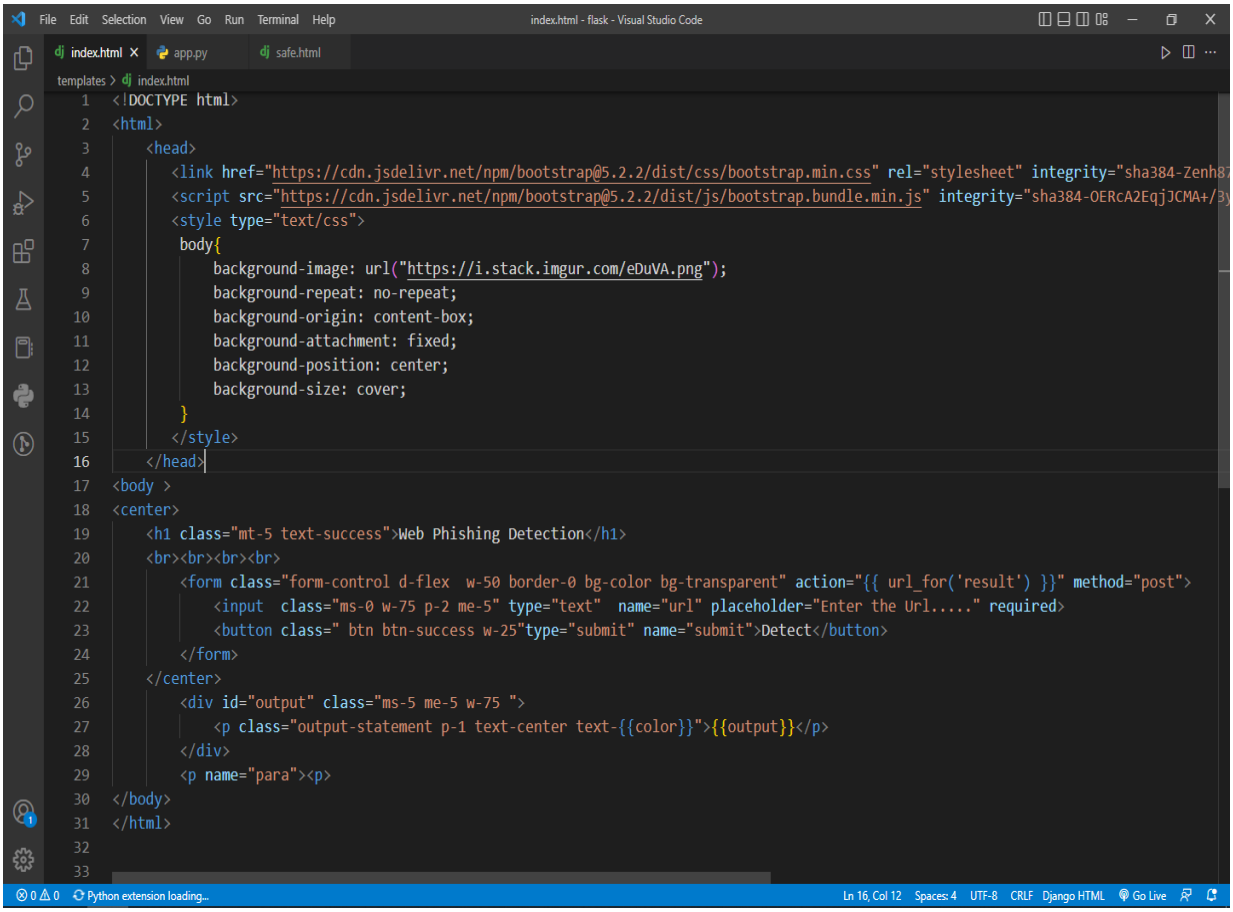


Model Building:

```
In [ ]: from sklearn.model_selection import train_test_split
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 50)

In [ ]: from sklearn.tree import DecisionTreeClassifier
        dc = DecisionTreeClassifier()
        dc_tree = dc.fit(x_train, y_train)

In [ ]: predict_dctree=dc_tree.predict(x_test)

In [ ]: accuracy(y_test, predict_dctree)

In [ ]: from sklearn.ensemble import RandomForestClassifier
        rf = RandomForestClassifier()
        predict_rf = rf.fit(x_train, y_train)

In [ ]: nzg=predict_rf.predict(x_test)

In [ ]: accuracy(y_test,nzg)

In [26]: with open('rm_forest__new_model','wb') as f:
           pickle.dump(predict_rf,f)
```

User Interface:

To interact with the user there, exist a user interface which allows the user to enter the "URL" into the input field

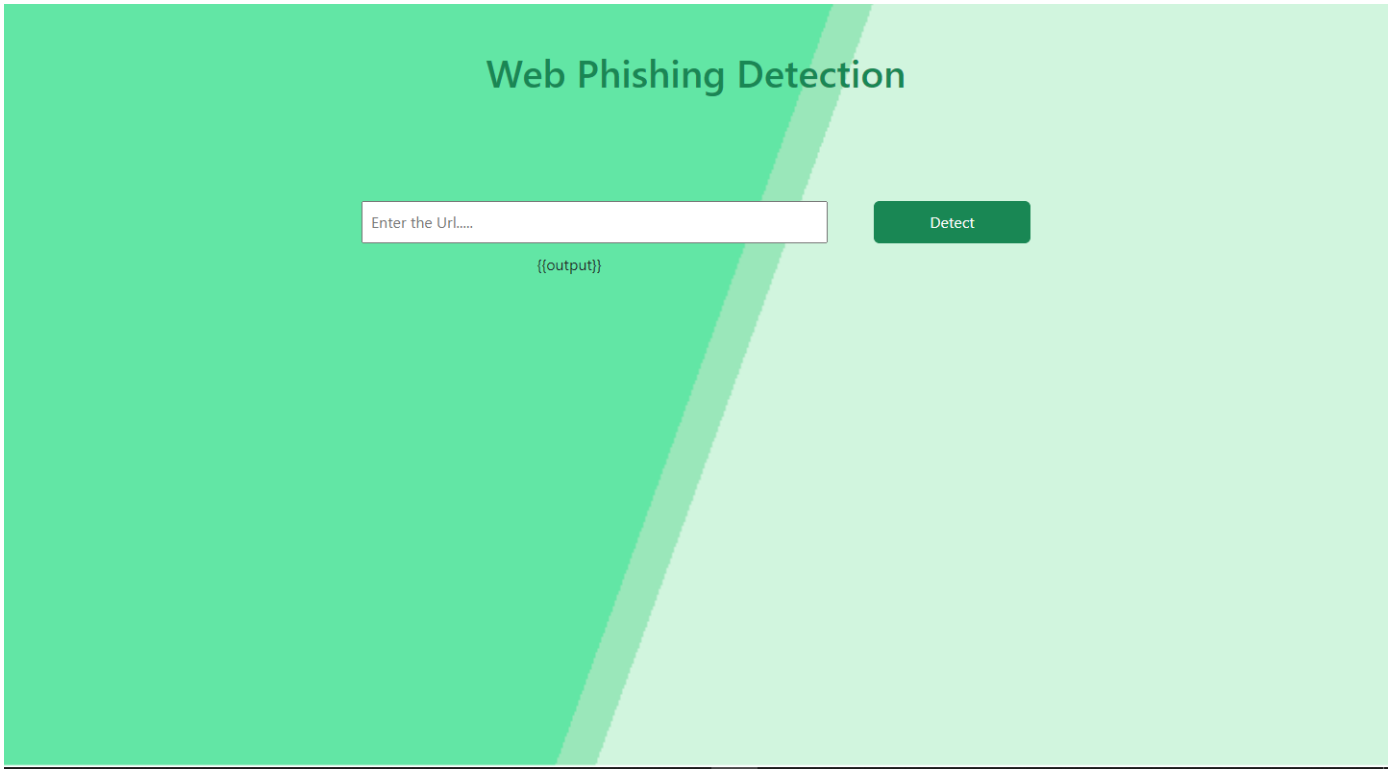

File Edit Selection View Go Run Terminal Help
index.html - flask - Visual Studio Code
index.html
app.py
safe.html
templates > index.html
<!DOCTYPE html>
<html>
<head>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-OERcA2EqjJCMA+/3y
<style type="text/css">
body{
background-image: url("https://i.stack.imgur.com/eDuVA.png");
background-repeat: no-repeat;
background-origin: content-box;
background-attachment: fixed;
background-position: center;
background-size: cover;
}
</style>
</head>
<body >

<h1 class="mt-5 text-success">Web Phishing Detection</h1>

<form class="form-control d-flex w-50 border-0 bg-color bg-transparent" action="{{ url_for('result') }}" method="post">
<input class="ms-0 w-75 p-2 me-5" type="text" name="url" placeholder="Enter the Url....." required>
<button class=" btn btn-success w-25"type="submit" name="submit">Detect</button>
</form>

{{output}}

</body>
</html>
Python extension loading...
Ln 16, Col 12 Spaces: 4 UTF-8 CRLF Django HTML Go Live


Web Phishing Detection
Enter the Url.....
Detect
{{output}}

Flask Integration:

This python file is responsible for the integration of the model that was earlier built-up with the Jupiter notebook and the User interface where the user will interact with the model

Code:

```
from flask import Flask, request, render_template, flash
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
import os
import scipy.io as sio
import joblib
warnings.filterwarnings('ignore')
from features import FeatureExtraction
import requests


app = Flask(__name__)
app.secret_key = "123abc$#@!"


API_KEY = "zlMpO3s-HVl2y64bgG00kmKBRH9SuJ_x5mCqgjF0sqYa"
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.route("/", methods=["GET", "POST"])
def index():
   return render_template("index.html")
@app.route("/result",methods=['POST','GET'])
def result():
   if request.method == "POST":

      url = request.form["url"]
      obj = FeatureExtraction(url)
      x = np.array(obj.getFeaturesList()).reshape(1,30)
      y=x.tolist()
      payload_scoring = {"input_data": [{"field": [[ 'having_IPhaving_IP_Address', 'URLURL_Length',
     'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting',
     'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State',
     'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token',
     'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
     'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover',
     'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord',
     'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page',
     'Statistical_report']], "values": y}]}



      response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/f6e86133-3fbb-484f-bfba-
d43e02ac57ad/predictions?version=2022-11-17', json=payload_scoring,
```

```
        headers={'Authorization': 'Bearer ' + mltoken})
        print('scoring response')
        predictions=response_scoring.json()
        pred=predictions['predictions'][0]['values'][0][0]
        print(pred)

        if pred == -1:
            pred="faileure"
            cond="UnSafe"
            color="danger"
        else:
            pred="success"
            cond="Safe"
            color="info"

        output = f"{pred} ! The entered URL/Link is {cond} to use"
        return render_template('index.html', output=output,color=color)

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

.

CHAPTER 8

TESTING

8.1 Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
Dataset_TC_OO1	**dataset**	Data	Firstly, Collect all the necessary datasets for the project	1.Login into the IBM dashboard 2.Navigate to the guided projects tab 3.Download the required dataset	Sample values from dataset	Dataset should be downloaded successfully	Working as expected	Pass
Preprocessing_TC_OO2	Dataset	Data	To clean the dataset like without any null values	1.Open google Collab 2.Import the required packages 3.Eliminate unwanted data	Sample value from dataset	No of the cell In the data set should be empty and no unwanted rows	Working as expected	Pass
Exploratory data analysis_TC_OO3	Dataset	Data	Verify whether ther is any unwanted data columns inn the data set	1.To load the data set 2.To eliminate the unwanted columns	Column should be without unwanted columns	No column without use or data without affecting the model	Working as expected	Pass
Model building_TC_OO4	Functional	Machine learnig model	To built the model to predict the provided data	1.Construct the basic work of the model 2.Add Dense layer for training the model 3.Save the model and train it	Dataset	Accuracy Over 90%	Working as expected	Pass
UI design_TC_OO5	Functional	Flask,HTML	To design an user interface flexible to the user	1. create the web page using html 2.Style the page using css or bootstrap	Web page	Good UI design	Working as expected	Pass

xxxxx UI Integration_TC_OO6	Functional	Flask	To Integrate the developed model with the created User interface	1. create a flask app 2. Import the model into the flask app 3.Import the Html file and iinntegrate both	Model and html pages	Integration of model with UI	Working as expected	Pass
Result_TC_OO6	Functional	Result	After identifying the result we shall display the result in thee same or new html page	1.Input will be given as link and split the url into tockens 2.Run the model 3.Find and display the result to the user	Dataset	Display of result	Working as expected	Pass

8.2 User Acceptance Testing:

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	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

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

CHAPTER 9

RESULTS

9.1 Performance metrics:

The median efficiency is used to assess each categorization model's effectiveness. The final item will appear in the way it was envisioned. Graphical representations are used to depict information during classification. The percentage of predictions made using the testing dataset is used to gauge accuracy. By dividing the entire number of forecasts even by properly predicted estimates, it is simple to calculate. The difference between actual and anticipated output is used to calculate accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positives, TN = True Negatives, FN = False Negatives and FP = False Positives.

Thus, accuracy for all the four used models were calculated and ranked. Random forest performed betterthan other models.

Fig 9.1 Performance metrics

Algorithm	Accuracy
Decision tree	95%
Random forest	97.14%

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- **Increases User Security** When the user enters the website he was motivated to redirect to some other pages or some other phishing activities .This can be avoided in this model .the model is built up with lots of data sets so that the accuracy tends to be high.

DISADVANTAGES:

- Can help the use only through the website and not dynamically whenever they click on a phishing website

∴

CHAPTER 11

CONCLUSION

There are many methods to perform phishing detection. Our system aims to enhance the detection method to detect phishing websites using machine learningtechnology.
This paper aims to enhance detection method to detect phishing websites using machine learning technology. We achieved 97.14% detection accuracy using random forest algorithm with lowest false positive rate. Also result shows that classifiers give better performance when we used more data as training data.

CHAPTER-12

FUTURE SCOPE

This issue can be solved by training a model with various types of algorithms I that some algorithms resembles

the others in the accuracy and some deviate here we built up the model with random forest algorithm ,but this model can be built up with the combination of various machine learning models like Naïve bayes ,random forest

Support vector machine ,decision tree etc.. and the model built In this manner will have more accuracy

In future hybrid technology will be implemented to detect phishing websites more accurately, for which random forest algorithm of machine learning technology and blacklist method will be used

CHAPTER 13

APPENDIX

13.1 Source code:

app.py

```
from flask import Flask, request, render_template, flash
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
import os
import scipy.io as sio
import joblib
warnings.filterwarnings('ignore')
from features import FeatureExtraction
import requests


app = Flask(__name__)
app.secret_key = "123abc$#@!"


API_KEY = "zlMpO3s-HVl2y64bgG00kmKBRH9SuJ_x5mCqgjF0sqYa"
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.route("/", methods=["GET", "POST"])
def index():
    return render_template("index.html")
@app.route("/result",methods=['POST','GET'])
def result():
    if request.method == "POST":
```

```
        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        y=x.tolist()
        payload_scoring = {"input_data": [{"field": [[ 'having_IPhaving_IP_Address',
'URLURL_Length',
       'Shortining_Service', 'having_At_Symbol', 'double_slash_redirecting',
       'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State',
       'Domain_registeration_length', 'Favicon', 'port', 'HTTPS_token',
       'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH',
       'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover',
       'RightClick', 'popUpWidnow', 'Iframe', 'age_of_domain', 'DNSRecord',
       'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page',
       'Statistical_report']], "values": y}]}


        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/f6e86133-3fbb-484f-bfba-
d43e02ac57ad/predictions?version=2022-11-17', json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        print('scoring response')
        predictions=response_scoring.json()
        pred=predictions['predictions'][0]['values'][0][0]
        print(pred)


        if pred == -1:
            pred="faileure"
            cond="UnSafe"
            color="danger"
        else:
            pred="success"
            cond="Safe"
            color="info"


        output = f"{pred} ! The entered URL/Link is {cond} to use"
        return render_template('index.html', output=output,color=color)
```

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

features.py

```
from urllib.parse import urlparse
import ipaddress
import re
import requests
```

```
import whois
from datetime import datetime

class FeatureExtraction:
    features=[]
    def_init_(self,url):
        self.features=[]
        self.url = url

        #Address bar based features
        self.features.append(self.having_IPhaving_IP_Address())
        self.features.append(self.URLURL_Length())
        self.features.append(self.Shortining_Service())
        self.features.append(self.having_At_Symbol())
        self.features.append(self.double_slash_redirecting())
        self.features.append(self.Prefix_Suffix())
        self.features.append(self.HTTPS_token())

        # HTML & Javascript based features
        try:
            self.response = requests.get(url)
        except:
            self.response = ""

        self.features.append(self.on_mouseover())
        self.features.append(self.RightClick())
        self.features.append(self.popUpWidnow())
        self.features.append(self.Iframe())

        #Domain based features
        dns = -1
        try:
            self.domain_name = whois.whois(urlparse(url).netloc)
        except:
```

```
            dns = 1


        self.features.append(1 if dns == 1 else self.age_of_domain())
        self.features.append(dns)


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


    # 2.longUrl
    def URLURL_Length(self):
        #print("Length")
        if len(self.url) < 54:
            return 1
        else:
            return -1


    # 3.shortUrl
    def Shortining_Service(self):
        #print("short")
        shortening_services =
r"bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is
\.gd|cli\.gs|" \

r"yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|s
nipurl\.com|" \

r"short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fi
c\.kr|loopt\.us|" \
```

```
r"doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.
co|lnkd\.in|db\.tt|" \

r"qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity
\.im|q\.gs|is\.gd|" \

r"po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls
\.org|x\.co|" \

r"prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com
|tweez\.me|v\.gd|" \
                    r"tr\.im|link\.zip\.net"
        match=re.search(shortening_services,self.url)
        if match:
            return -1
        else:
            return 1


    # 4.Symbol@
    def having_At_Symbol(self):
        #print("at")
        if "@" in self.url:
            return -1
        else:
            return 1


    # 5.Redirecting//
    def double_slash_redirecting(self):
        #print("//")
        pos = self.url.rfind('//')
        if pos > 6:
            if pos > 7:
                return -1
            else:
                return 1
        else:
            return 1
```

```
# 6.prefixSuffix
def Prefix_Suffix(self):
    #print("prefix")
    if '-' in urlparse(self.url).netloc:
        return -1
    else:
        return 1

#HTTPS token
def HTTPS_token(self):
    #print("https")
    domain = urlparse(self.url).netloc
    if 'https' in domain:
        return -1
    else:
        return 1

def on_mouseover(self):
    #print("mouse")
    try:
        if re.findall("", self.response.text):
            return -1
        else:
            return 1
    except:
         return -1

def RightClick(self):
    #print("right")
    if self.response == "":
        return -1
    else:
        if re.findall(r"event.button ?== ?2", self.response.text):
```

```
                return 1
            else:
                return -1

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

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

    # 13.Survival time of domain: The difference between termination time
and creation time (Domain_Age)
    def age_of_domain(self):
        #print("age")
        creation_date = self.domain_name.creation_date
        expiration_date = self.domain_name.expiration_date
        if (isinstance(creation_date,str) or
isinstance(expiration_date,str)):
            try:
```

```
                creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
                expiration_date = datetime.strptime(expiration_date,"%Y-%m-
%d")
            except:
                return -1
        if ((expiration_date is None) or (creation_date is None)):
            return -1
        elif ((type(expiration_date) is list) or (type(creation_date) is
list)):
            return -1
        else:
            ageofdomain = abs((expiration_date - creation_date).days)
            if ((ageofdomain/30) < 6):
                return -1
            else:
                return 1
```

```
    def getFeaturesList(self):
        print(self.features)
        return self.features
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
       <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
       <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
       <style type="text/css">
        body{
            background-image: url("https://i.stack.imgur.com/eDuVA.png");
            background-repeat: no-repeat;
            background-origin: content-box;
            background-attachment: fixed;
            background-position: center;
            background-size: cover;
        }
       </style>
    </head>
<body >
<center>
    <h1 class="mt-5 text-success">Web Phishing Detection</h1>
    <br><br><br><br>
        <form class="form-control d-flex  w-50 border-0 bg-color bg-transparent"
action="{{ url_for('result') }}" method="post">
            <input  class="ms-0 w-75 p-2 me-5" type="text"  name="url"
placeholder="Enter the Url....." required>
            <button class=" btn btn-success w-25"type="submit"
name="submit">Detect</button>
        </form>
```

```
    </center>
        <div id="output" class="ms-5 me-5 w-75 ">
            <p class="output-statement p-1 text-center text-{{color}}">{{output}}</p>
        </div>
        <p name="para"><p>
</body>
</html>
```

13.2 GitHub & project demo link:

GitHub link: IBM-EPBL/IBM-Project-14297-1659548839: Web Phishing Detection (github.com)

Demo link: HookPhish - Web Phishing Detector - YouTube