PROJECT REPORT

Project Title - Web Phishing Detection using Machine Learning

Team Members

- Mahath PT (Team Leader)
- Janarthanan M
- Habib Raja PA
- Meer Abzal Hussain A

INDEX

1.INTRODUCTION

1. Project Overview

A phishing attack is one of the most significant problems faced by online users because of its enormous effect on the online activities performed. In recent years, phishing attacks continue to escalate in frequency, severity and impact. Several solutions, using various methodologies, have been proposed in the literature to counter the web-phishing threats. Notwithstanding, the existing technology cannot detect the new phishing attacks accurately due to the insufficient integration of features of the text, image and frame in the evaluation process. The use of related features of images, frames and text of legitimate and non-legitimate websites and associated artificial intelligence algorithms to develop an Integrated method to address these together. This paper presents an Adaptive Neuro-Fuzzy Inference System (ANFIS) based robust scheme using the integrated features of the text, images and frames for web-phishing detection and protection. The proposed solution achieves 98.3% accuracies. To our best knowledge, this is the first work that considers the best-integrated text, image and frame feature based solution for phishing detection scheme.

2. Purpose

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 measures and compared.

2.LITERATURE SURVEY

1. Existing solution

we introduce the proposed intelligent phishing detection and protection system (IPDPS). We also look at the different issues that arise in detecting phishing websites. This section covers phishing detection implementation using sets of the dataset , the essential characteristics and features of phishing website extraction techniques. Developing with the

anti-phishing methods, phishers use various phishing methods and more complex and hard-to-detect approaches. The most straightforward way for a phisher to swindle people is to make the phishing web page similar to their target. However, many distinctive and features can distinguish the original legitimate website from the clone phishing website like the spelling error, image alteration, long URL address and abnormal DNS records. The full list is revealed in Table 3 which is used later in our analysis and classification study. If an attacker clones a legitimate website as a whole or designed to look similar as they usually do in most attacks in recent times, our approach is that similar looking phishing web page content is not left for the users to check for the indicator or the authenticity attentively, but can detect by automated methods. Our approach is based on website phishing detection using the features of the site, content and their appearance. These properties are stored in a local database (Excel table) as a knowledge model and first compared with the newly loaded site at the time of loading against the dangerous web page offline. After the comparison was unable to detect the similarity, then the critical approach to compare the legitimate and fake using the features of the website with machine learning for an intelligent decision.

In this study, reviewing different phishing investigations, research papers, conducting a separate phishing experimental case study give us more insight into the selection of the feature used for our phishing classification. Given this, we can extract 35 elements and factors which characterise the signature of any phishing website incident. These datasets divided into seven criteria that are distributed into three layers, depending on the attack type. The most popular feature selection methods in the literature are ChiSquare (x 2) and information gain. Chi-Square is the commonly used method and adopted in this study; evaluated features are by computing Chi-Square statistics on classes (Gaunt, 2016). The information gain technique is also used in feature selection, which decreases the size of features by calculating the value of each attribute and ranking them. In other words, information gain selects elements through scores (Zeng et al., 2016). In this approach, a subset of initially chosen features is only used for testing and training the classifier (Abunadi et al., 2013). Feature extraction usually converts the original feature space into a more compact space. However, the original features are retained and transformed into a new reduced space with only a few representative sets (Zareapoor and Seeja, 2015). This approach mainly uses principal component analysis (PCA) and latent semantic analysis (LSA). Principal component analysis reduces the dimension of the data by transforming the actual attribute space into a smaller one (Vidal et al., 2016). That could achieve by converting the real variables = [y1, y2, … , yn] (where n is n tan h number of actual variables) into new variables, = [t1,t2, … ,tp] (where p is an n th number of the new set of variables). The LSA technique is a novel-based method of text classification. The approach analyses the relationship between a concept and term contained in unstructured data, and it has the ability to correlate semantically related value that are latent in nature (Marcolin and Becker, 2016). These processes are used to convert the hybrid features into data that can be used to train and test our model.

2.References

1. Alkhozae, M. G. and Batarfi, O. A. (2011) 'Phishing websites detection based on phishing characteristics in the web page source code', International Journal of Information and Communication Technology Research, 1(6).

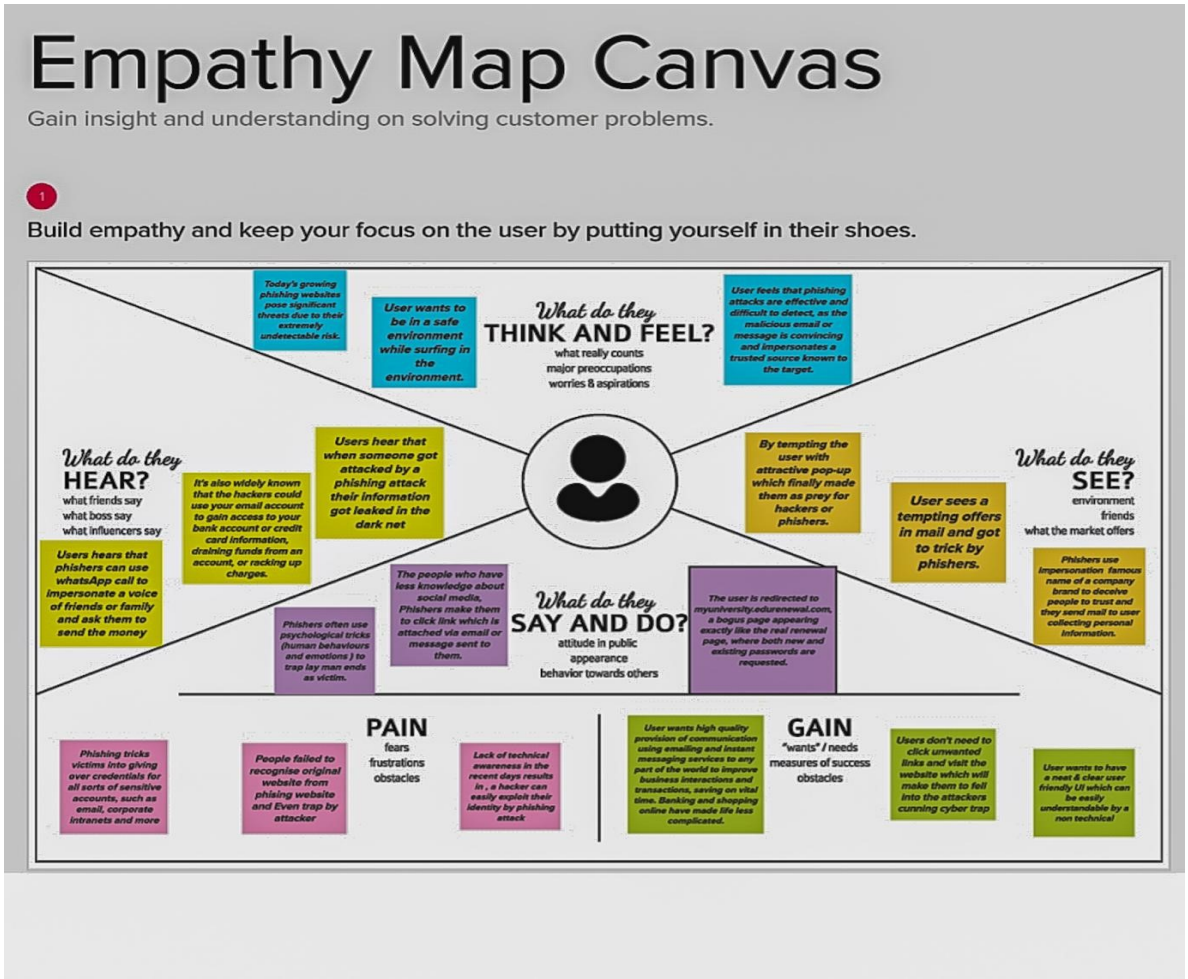
2. Aburub, and S. Alhawari, \"A new fast associative classification algorithm for detecting phishing websites,\" Appl. Soft Comput.. [5] \"Multi-label rules for phishing classification,\" Appl. Comput. Informatics, vol. 11, no. 1, pp. 29–46, 2015.

3. Problem Statement Definition

‘Phishing sites’ are some type of the internet security issues that mainly targets the human vulnerabilities compared to software vulnerabilities. Phishing sites are malicious websites that imitate as legitimate websites or web pages and aim to steal user’s personal credentials like user id, password, and financial information. Spotting these phishing websites is typically a challenging task because phishing is mainly a semantics-based attack, that mainly focus on human vulnerabilities, not the network or software vulnerabilities. Phishing can be elaborated as the process of charming users in order to gain their personal credentials like user-id’s and passwords. In this paper, we come up with an intelligent system that can spot the phishing sites. This intelligent system is based on a machine learning model. Our aim through this paper is to stalk a better performance classifier by examining the features of the phishing site and choose appropriate combination of systems for the training of the classifier.

3. IDEATION AND PROPOSED SOLUTION

1. empathy map canvas



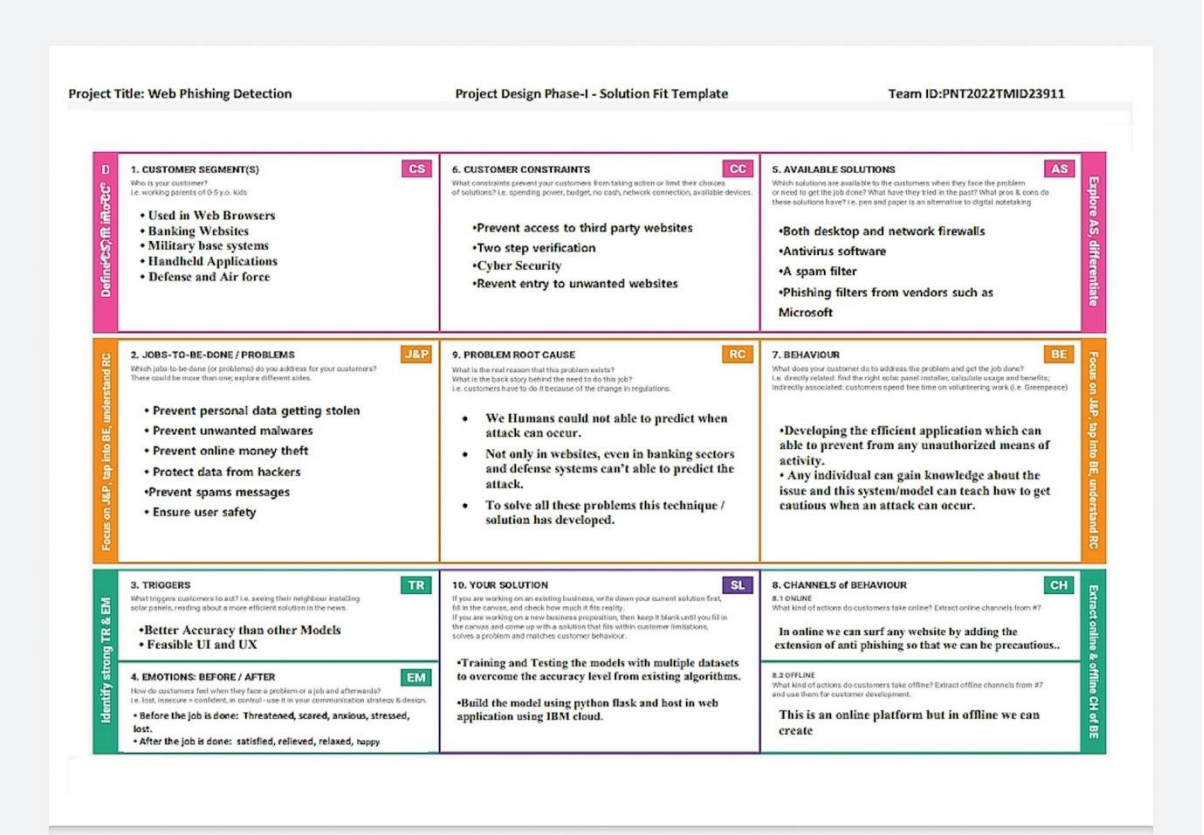
2. Ideation & Brainstorming

3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	1. Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. 2. Typically a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites in order to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details.
2.	Idea / Solution description	1. The phishing website can be detected based on some important characteristics like URL and Domain Identity, and security and encryption criteria in the final phishing detection rate. 2. Once user makes transaction through online when he makes

		payment through the website our system will use data mining algorithm to detect whether the website is phishing website or not. 3.This application can be used by many Ecommerce enterprises in order to make the whole transaction process secure. 4.Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms. With the help of this system user can also purchase products online without any hesitation. 5. Admin can add phishing website url or fake website url into system where system could access and scan the phishing website and by using algorithm, it will add new suspicious keywords to database. System uses machine learning technique to add new keywords into database.
3.	Novelty / Uniqueness	1. Machine learning technology consists of a many algorithms which requires past data to make a decision or prediction on future data 2. Using this technique, algorithm will analyze various blacklisted and legitimate URLs and their features to accurately detect the phishing websites including zero- hour phishing websites.
4.	Social Impact / Customer Satisfaction	From every phishing incident that has ever taken place in history, one constant effect is financial loss. First is the direct loss from transferred funds by employees who were fooled by the hackers. Second is the fines for non-compliance imposed by regulatory bodies like HIPAA, PCI, and PIPEDA, among others
5.	Business Model (Revenue Model)	People buy subscription annually, to protect their files both locally and at remote location with the help of our cloud integrated flask app for web phishing detection.
6.	Scalability of the Solution	1. We deliver the Good feasible UI/UX design on Web phishing detection. 2. The model is tested and trained in multiple types of datasets to get high accuracy than other algorithms

3. Problem Solution Fit



4. REQUIREMENT ANALYSIS

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 Google form Registration through Gmail Registration through whatsapp Registration through Facebook
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Extraction and prediction	It retrieves features based on heuristics and visual similarities. The URL is predicted by the model using Machine Learning methods such as Logistic Regression and KNN.
FR-4	Real time monitoring	The use of Extension plugin should provide a warning pop-up when they visit a website that is phished. Extension plugin will have the capability to also detect latest and new phishing websites
FR-5	Authentication	Authentication assures secure site, secure processes and enterprise information security

2. Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Engage the user about the process to ensure that the functionality can meet design and usability requirements
NFR-2	**Security**	It guarantees that any data included within the system or its components will be safe from malware threats or unauthorised access. If you wish to prevent unauthorised access to the admin panel, describe the login flow and different user roles as system behaviour or user actions.
NFR-3	**Reliability**	It focuses on preventing failures during the lifetime of the product or system, from commissioning to decommissioning.
NFR-4	**Performance**	It is concerned with a measurement of the system's reaction time under various load circumstances
NFR-5	**Availability**	Ensuring that the application can meet its availability targets to be resilient (fault tolerance)
NFR-6	**Scalability**	It has access to the highest workloads that will allow the system to satisfy the performance criteria. There are two techniques to enable the system to grow as workloads increase: Vertical and horizontal scaling

5. PROJECT DESIGN

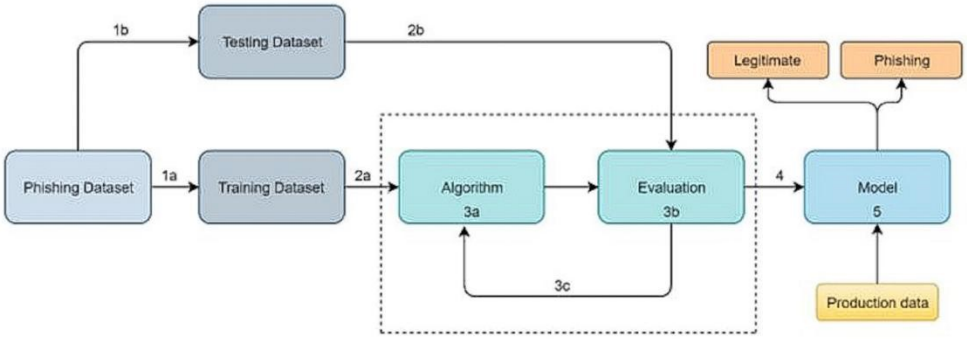
1. Data Flow Diagram

Project Design Phase-II
Data Flow Diagram & User Stories

Date	03 October 2022
Team ID	PNT2022TMID23911
Project Name	Project – Web Phishing Detection
Maximum Marks	4 Marks

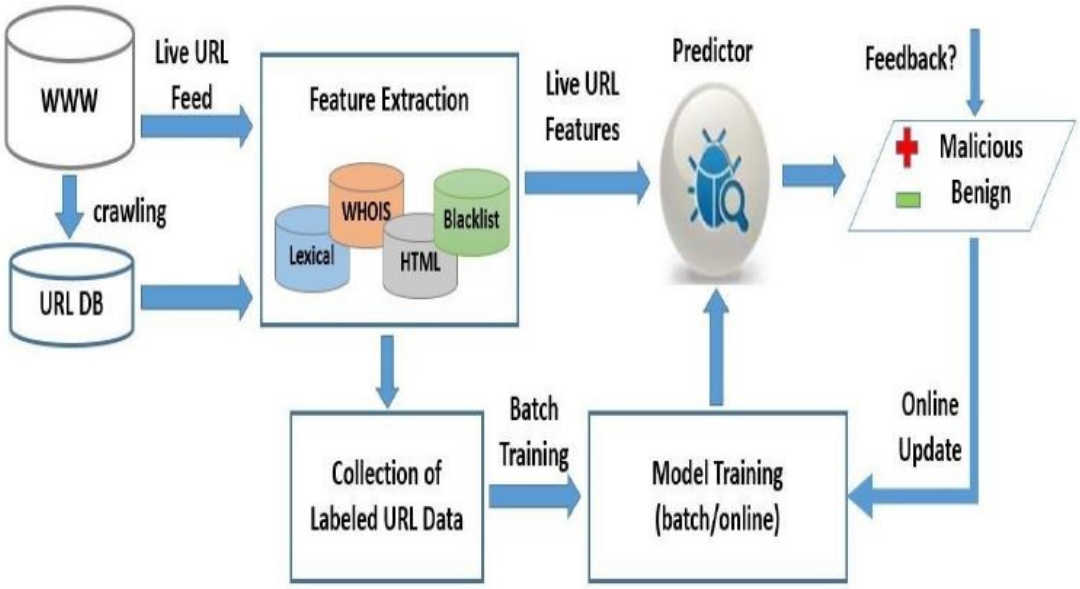
Data Flow Diagrams:

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



2. SOLUTION AND TECHNICAL ARCHITECTURE

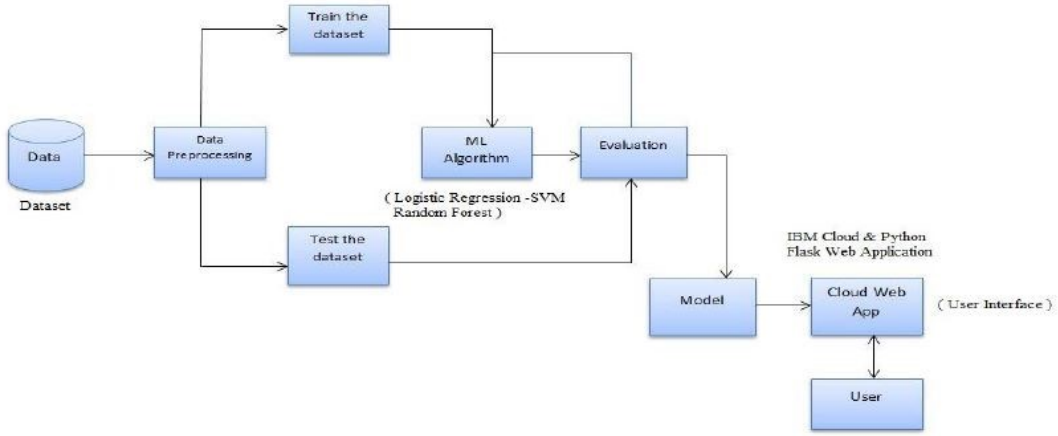
Example - Solution Architecture Diagram:



Solution Architecture

Technical Architecture:

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



Technical Architecture

3. User Stories

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (Web 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
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm
		USN-3	As a user, I can register for the application through facebook	I can register & access the dashboard with facebook Login
		USN-4	As a user, I can register for the application through Gmail	
	Login	USN-5	As a user, I can log into the application by entering email & password	
	Dashboard			
Customer(web user)	User input	USN-1	As a user I can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem.
Customer care executive	Feature extraction	USN-1	After I compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a User I can have comparison between websites for security.

Administrator	prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN.	In this I can have correct prediction on the particular algorithms.
	classifier	USN-2	Here i will send all the model output to classifier in order to produce final result.	this I will find the correct classifier for producing the result .

6. PROJECT PLANNING AND SCHEDULING

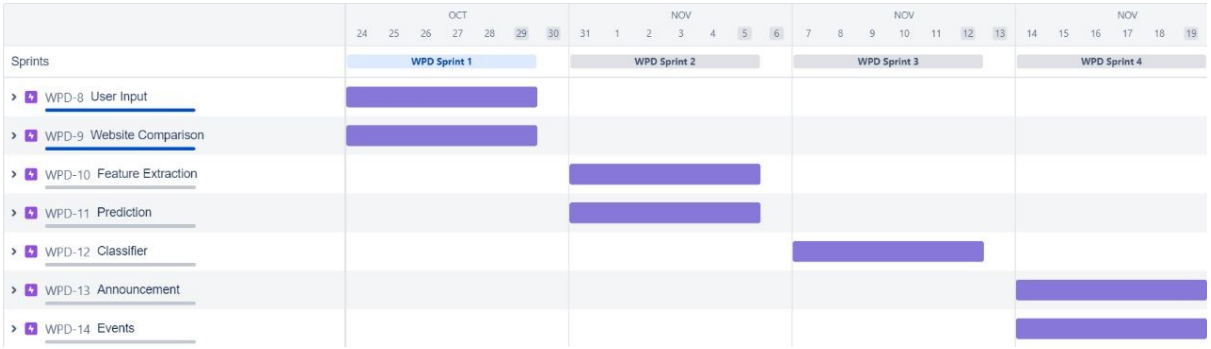
1. Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation	1	Medium	Janarthanan M
Sprint-1	Website comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach	1	High	Mahath PT
Sprint-2	Feature extraction	USN-3	After comparison, if none found on comparison then it extracts feature using heuristic and visual similarity	2	High	Meer abzal hussain A
Sprint-2	prediction	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	1	Medium	Habib raja PA
Sprint-3	classifier	USN-5	Model then displays whether the website is legal site or a phishing site	1	Medium	Janarthanan M
Sprint-4	announcement	USN-6	Model then displays whether the website is legal site or a phishing site	1	High	Habib raja PA
Sprint-4	events	USN-7	This model needs the capability of retrieving and displaying accurate result for a website.	1	High	Mahath PT

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	20	6 Days	05 Nov 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

3. Reports for JIRA



7. CODING AND SOLUTIONING

1. Feature 1

```
1 #app.py
 # importing required libraries
 from feature import FeatureExtraction


 from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
 import pickle
warnings.filterwarnings('ignore')
file = open("model.pkl", "rb")
```

```
 gbc = pickle.load(file)
file.close()


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


        url = request.form["url"]
        obj = FeatureExtraction(url)
       x = np.array(obj.getFeaturesList()).reshape(1, 30)

       y_pred = gbc.predict(x)[0]
       #1 is safe
       #-1 is unsafe
       y_pro_phishing = gbc.predict_proba(x)[0, 0]
       y_pro_non_phishing = gbc.predict_proba(x)[0, 1]
       # if(y_pred ==1 ):
       pred = "Itis {0:.2f} % safe to go ".format(y_pro_phishing*100)
       return render_template('index.html', xx=round(y_pro_non_phishing, 2), url=url)
return render_template("index.html", xx=-1)


if __name__ == " main ":
       app.run(debug=True, port=2002)
```

2. Feature 2

```
#feature.py
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
importsocket
importrequests
from googlesearch importsearch
import whois
from datetime import date, datetime
importtime
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse


class FeatureExtraction:
       features = []
       def__ init (self, url):
       self.features = []
       self.url = url
```

```
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""
try:
        self.response = requests.get(url)
        self.soup = BeautifulSoup(response.text, 'html.parser')
except:
        pass
try:
        self.urlparse = urlparse(url)
        self.domain = self.urlparse.netloc
        except:
        pass
try:
        self.whois_response = whois.whois(self.domain)
except:
        pass
        self.features.append(self.UsingIp())
        self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())

self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
self.features.append(self.LinksPointingToPage())
```

```
self.features.append(self.StatsReport())

# 1.UsingIp

def UsingIp(self):
try:
        ipaddress.ip_address(self.url)
        return -1
except:
        return 1
# 2.longUrl

def longUrl(self):
        if len(self.url) < 54:
                return 1
        if len(self.url) >= 54 and len(self.url) <= 75:
                return 0
        return -1

# 3.shortUrl
defshortUrl(self):
match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|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',self.url)
        if match:
                return -1
        return 1

# 4.Symbol@
defsymbol(self):
        if re.findall("@",self.url):
                return -1
        return 1

# 5.Redirecting//
def redirecting(self):
        ifself.url.rfind('//') > 6:
                return -1
```

```
        return 1

# 6.prefixSuffix
def prefixSuffix(self):
        try:
        match = re.findall('\-', self.domain)
        if match:
                return -1
        return 1
except:
        return -1

# 7.SubDomains
def SubDomains(self):
        dot_count = len(re.findall("\.",self.url))
        if dot_count == 1:
                return 1
        elif dot_count == 2:
                return 0
        return -1

# 8.HTTPS
defHppts(self):
        try:
                https = self.urlparse.scheme
                if 'https' in https:
                        return 1
        return -1
except:
        return 1

# 9.DomainRegLen
def DomainRegLen(self):
        try:
                expiration_date = self.whois_response.expiration_date
                creation_date = self.whois_response.creation_date
                try:
                        if(len(expiration_date)):
                                expiration_date = expiration_date[0]
        except:
                pass
        try:
                if(len(creation_date)):
                creation_date = creation_date[0]
except:
        pass
        age = (expiration_date.year-creation_date.year)*12 + \
                (expiration_date.month-creation_date.month)
        if age >= 12:
                return 1
```

```
        return -1
except:
        return -1

# 10. Favicon
def Favicon(self):
        try:
                for head in self.soup.find_all('head'):
                        for head.link in self.soup.find_all('link', href=True):
                                dots = [x.start(0)
                                        for x in re.finditer('\.', head.link['href'])]
                                if self.url in head.link['href'] or len(dots) == 1 or domain in
head.link['href']:
                                        return 1
        return -1
except:
        return -1

# 11. NonStdPort
def NonStdPort(self):
        try:
                port = self.domain.split(":")
                iflen(port) > 1:
                        return -1
        return 1
except:
        return -1

# 12. HTTPSDomainURL
def HTTPSDomainURL(self):
        try:
                if 'https' in self.domain:
                        return -1
                return 1
        except:
                return -1

# 13. RequestURL
def RequestURL(self):
        try:
                for img in self.soup.find_all('img',src=True):
                        dots = [x.start(0) for x in re.finditer('\.', img['src'])]
                        ifself.url in img['src'] or self.domain in img['src'] orlen(dots) == 1:
                                success = success + 1
                i = i+1
        for audio in self.soup.find_all('audio',src=True):
                dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
                ifself.url in audio['src'] orself.domain in audio['src'] or len(dots) == 1:
                        success = success + 1
                i = i+1
```

```
            for embed in self.soup.find_all('embed',src=True):
                dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
                ifself.url in embed['src'] orself.domain in embed['src'] orlen(dots) == 1:
                    success = success + 1
                i = i+1

            for iframe in self.soup.find_all('iframe',src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                ifself.url in iframe['src'] orself.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

# 14. AnchorURL
def AnchorURL(self):
        try:
            i, unsafe = 0, 0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in
a['href'].lower() or not(url
in a['href'] or self.domain in a['href']):
                    unsafe = unsafe + 1
            i = i + 1
        try:
            percentage = unsafe / float(i) * 100
            if percentage < 31.0:
                return 1
            elif ((percentage >= 31.0) and (percentage < 67.0)):
                return 0
            else:
                return -1
            except:
                return -1
        except
            return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
```

```
        try:
                i,success = 0, 0

                for link in self.soup.find_all('link', href=True):
                        dots = [x.start(0) for x in re.finditer('\.', link['href'])]
                        ifself.url in link['href'] orself.domain in link['href'] orlen(dots) == 1:
                                success = success + 1
                        i = i+1
                for scriptin self.soup.find_all('script',src=True):
                        dots = [x.start(0) for x in re.finditer('\.',script['src'])]
                        ifself.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
                                success = success + 1
                        i = i+1
        try:
                        percentage = success / float(i) * 100
                        if percentage < 17.0:
                                return 1
                        elif((percentage >= 17.0) and (percentage < 81.0)):
                                return 0
                        else:
                                return -1
                except:
                        return 0
        except:
                return -1

# 16. ServerFormHandler
def ServerFormHandler(self):
        try:
                if len(self.soup.find_all('form', action=True)) == 0:
                        return 1
        else:
                for form in self.soup.find_all('form', action=True):
                        if form['action'] == "" or form['action'] == "about:blank":
                                return -1
                        elifself.url notin form['action'] and self.domain notin form['action']:
                                return 0
                        else:
                                return 1
        except:
                return -1

# 17. InfoEmail
def InfoEmail(self):
        try:
                ifre.findall(r"[mail\(\)|mailto:?]",self.soap):
                        return -1
                else:
                        return 1
        except:
```

```
        return -1

# 18. AbnormalURL
def AbnormalURL(self):
        try:
                if self.response.text == self.whois_response:
                        return 1
                else:
                        return -1
        except:
                return -1

# 19. WebsiteForwarding
def WebsiteForwarding(self):
        try:
                if len(self.response.history) <= 1:
                        return 1
                elif len(self.response.history) <= 4:
                        return 0
                else:
                        return -1
        except:
                return -1

# 20. StatusBarCust
def StatusBarCust(self):
        try:
                if re.findall("<script>.+onmouseover.+</script>", self.response.text):
                        return 1
                else:
                        return -1
        except:
                return -1

# 21. DisableRightClick
def DisableRightClick(self):
        try:
                if re.findall(r"event.button ?== ?2", self.response.text):
                        return 1
                else:
                        return -1
        except:
                return -1

# 22. UsingPopupWindow
def UsingPopupWindow(self):
        try:
                if re.findall(r"alert\(", self.response.text):
                        return 1
                else:
```

```
                return -1
        except:
                return -1

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

# 24. AgeofDomain
def AgeofDomain(self):
        try:
                creation_date = self.whois_response.creation_date
                try:
                        if(len(creation_date)):
                                creation_date = creation_date[0]
                except:
                        pass

                today = date.today()
                age = (today.year-creation_date.year) * \
                        12+(today.month-creation_date.month)
                if age >= 6:
                        return 1
                return -1
        except:
                return -1

# 25. DNSRecording
def DNSRecording(self):
        try:
                creation_date = self.whois_response.creation_date
                try:
                        if(len(creation_date)):
                                creation_date = creation_date[0]
                except:
                        pass
                today = date.today()
                age = (today.year-creation_date.year) * \
                        12+(today.month-creation_date.month)
                if age >= 6:
                        return 1
                return -1
        except:
                return -1
```

```
# 26. WebsiteTraffic
def WebsiteTraffic(self):
	try:
		rank = BeautifulSoup(urllib.request.urlopen(
			"http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(),
"xml").find("REACH")['RANK']
		if (int(rank) < 100000):
			return 1
		return 0
	except:
		return -1

# 27. PageRank
def PageRank(self):
	try:
		prank_checker_response = requests.post(
			"https://www.checkpagerank.net/index.php", {"name":self.domain})
		global_rank = int(re.findall(
			r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
		if global_rank > 0 and global_rank < 100000:
			return 1
		return -1
	except:
		return -1

# 28. GoogleIndex
def GoogleIndex(self):
	try:
		site = search(self.url, 5)
		if site:
			return 1
		else:
			return -1
	except:
		return 1

# 29. LinksPointingToPage
def LinksPointingToPage(self):
	try:
		number_of_links = len(re.findall(r"<a href=", self.response.text))
		if number_of_links == 0:
			return 1
		elif number_of_links <= 2:
			return 0
		else:
			return -1
	except:
		return -1

# 30. StatsReport
```

```
def StatsReport(self):
	try:
		url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt
|ow\.ly', url)
	ip_address = socket.gethostbyname(self.domain)
	ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.21
1\.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|10
7\.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\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\
.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'
'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.1
56\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)

if url_match:
		return -1
	elifip_match:
		return -1
	return 1
except:
	return 1
def getFeaturesList(self):
	return self.features
```

8. TESTING

1. Test Cases

TESTCASES REPORT

				Date	05-Nov-22								
				Team ID	PNT2022TMID11486								
				Project Name	Project - Web Phishing Detection								
				Maximum Marks	4 marks								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	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 processing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		S Balaji
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements is Responsive		1.Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		R Abisheik
LoginPage_TC_OO3	Functional	Home page	Verify whether the link is legitimate or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		T S Aswin
LoginPage_TC_OO4	Functional	Home Page	Verify user is able to access the legitimate website or not		1.Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://phishing-shield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		Balajee A V
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 or be cautious if it is not secure	1. https://avbalajee.github.io/Awesome_2. totalpad.com 3. https://www.klince.edu 4. salesorion.info 5. https://www.google.com/ 6. delpets.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		Balajee A V

2. User Acceptance Testing

2. 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	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	70

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

9. RESULTS

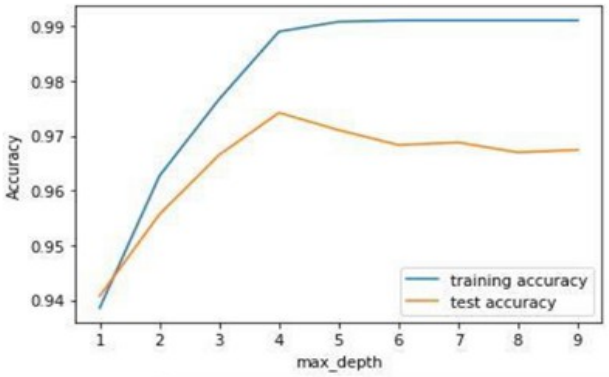
1. Performance Metrics

1. METRICS:
CLASSIFICATION REPORT:

```
In [52]: #computing the classification report of the model

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

```
              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

    accuracy                           0.97      2211
   macro avg       0.98      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

PERFORMANCE :



Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

10. ADVANTAGES AND DISADVANTAGES

Advantages

This system can be used by many E-commerce or other websites in order to have good customer relationship. User can make online payment securely. Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms With the help of this system user can also purchase products online without any hesitation.

Disadvantages

If Internet connection fails, this system won't work.
All websites related data will be stored in one place

11. CONCLUSION & FUTURE WORKS

This paper presented an intelligent phishing detection and protection scheme by employing a new approach using the integrated features of images, frames and text of phishing websites. An efficient ANFIS algorithm was developed, tested and verified for phishing website detection and protection based on the schemes proposed in Aburrous et al. (2010) and Barraclough et al. (2015). A set of experiments was performed using 13,000 available datasets. The approach showed an accuracy of 98.3%, which so far, is the best-integrated solutions for web-phishing detection and protection. The primary contribution of this study is the integration of hybrid features that have been extracted from text, images and frames and that are then used to develop a robust ANFIS solution. Future work will include using another algorithm like deep-learning for phishing web page detection and compare the effectiveness with the current result. More also, a web browser plug-in will be developed based on an efficient algorithm to detect phishing website and thus protect users in real time.

12. APPENDIX

SOURCE CODE

App.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("Web Phishing Detection.pkl","rb")
gbc = pickle.load(file)
file.close()

app=Flask(__name__)

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

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
```

```
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url )
    return render_template("index.html", xx =-1)


if __name__ == '__main__':
    app.run(debug=True,port=(2025))
```

Feature.py

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

class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass

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

        try:
            self.whois_response = whois.whois(self.domain)
        except:
            pass




        self.features.append(self.UsingIp())
```

```
        self.features.append(self.longUrl())
        self.features.append(self.shortUrl())
        self.features.append(self.symbol())
        self.features.append(self.redirecting())
        self.features.append(self.prefixSuffix())
        self.features.append(self.SubDomains())
        self.features.append(self.Hppts())
        self.features.append(self.DomainRegLen())
        self.features.append(self.Favicon())


        self.features.append(self.NonStdPort())
        self.features.append(self.HTTPSDomainURL())
        self.features.append(self.RequestURL())
        self.features.append(self.AnchorURL())
        self.features.append(self.LinksInScriptTags())
        self.features.append(self.ServerFormHandler())
        self.features.append(self.InfoEmail())
        self.features.append(self.AbnormalURL())
        self.features.append(self.WebsiteForwarding())
        self.features.append(self.StatusBarCust())

        self.features.append(self.DisableRightClick())
        self.features.append(self.UsingPopupWindow())
        self.features.append(self.IframeRedirection())
        self.features.append(self.AgeofDomain())
        self.features.append(self.DNSRecording())
        self.features.append(self.WebsiteTraffic())
        self.features.append(self.PageRank())
        self.features.append(self.GoogleIndex())
        self.features.append(self.LinksPointingToPage())
        self.features.append(self.StatsReport())


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

    # 2.longUrl
    def longUrl(self):
        if len(self.url) < 54:
            return 1
        if len(self.url) >= 54 and len(self.url) <= 75:
            return 0
        return -1

    # 3.shortUrl
    def shortUrl(self):
        match =
re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                  'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|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', self.url)
        if match:
            return -1
        return 1

    # 4.Symbol@
    def symbol(self):
        if re.findall("@",self.url):
            return -1
        return 1

    # 5.Redirecting//
    def redirecting(self):
        if self.url.rfind('//')>6:
            return -1
        return 1

    # 6.prefixSuffix
    def prefixSuffix(self):
        try:
            match = re.findall('\-', self.domain)
            if match:
                return -1
            return 1
        except:
            return -1

    # 7.SubDomains
    def SubDomains(self):
        dot_count = len(re.findall("\.", self.url))
        if dot_count == 1:
            return 1
        elif dot_count == 2:
            return 0
        return -1

    # 8.HTTPS
    def Hppts(self):
        try:
            https = self.urlparse.scheme
            if 'https' in https:
                return 1
            return -1
        except:
            return 1

    # 9.DomainRegLen
```

```
    def DomainRegLen(self):
        try:
            expiration_date = self.whois_response.expiration_date
            creation_date = self.whois_response.creation_date
            try:
                if(len(expiration_date)):
                    expiration_date = expiration_date[0]
            except:
                pass
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

            age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-
creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1

    # 10. Favicon
    def Favicon(self):
        try:
            for head in self.soup.find_all('head'):
                for head.link in self.soup.find_all('link', href=True):
                    dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                    if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                        return 1
            return -1
        except:
            return -1

    # 11. NonStdPort
    def NonStdPort(self):
        try:
            port = self.domain.split(":")
            if len(port)>1:
                return -1
            return 1
        except:
            return -1

    # 12. HTTPSDomainURL
    def HTTPSDomainURL(self):
        try:
            if 'https' in self.domain:
                return -1
            return 1
        except:
            return -1

    # 13. RequestURL
```

```
    def RequestURL(self):
        try:
            for img in self.soup.find_all('img', src=True):
                dots = [x.start(0) for x in re.finditer('\.', img['src'])]
                if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for audio in self.soup.find_all('audio', src=True):
                dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
                if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for embed in self.soup.find_all('embed', src=True):
                dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
                if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for iframe in self.soup.find_all('iframe', src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

    # 14. AnchorURL
    def AnchorURL(self):
        try:
            i,unsafe = 0,0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not
(url in a['href'] or self.domain in a['href']):
                    unsafe = unsafe + 1
                i = i + 1

            try:
                percentage = unsafe / float(i) * 100
                if percentage < 31.0:
                    return 1
                elif ((percentage >= 31.0) and (percentage < 67.0)):
                    return 0
```

```
            else:
                return -1
        except:
            return -1

    except:
        return -1

# 15. LinksInScriptTags
def LinksInScriptTags(self):
    try:
        i,success = 0,0

        for link in self.soup.find_all('link', href=True):
            dots = [x.start(0) for x in re.finditer('\.', link['href'])]
            if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
                success = success + 1
            i = i+1

        for script in self.soup.find_all('script', src=True):
            dots = [x.start(0) for x in re.finditer('\.', script['src'])]
            if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
                success = success + 1
            i = i+1

        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                return 1
            elif((percentage >= 17.0) and (percentage < 81.0)):
                return 0
            else:
                return -1
        except:
            return 0
    except:
        return -1

# 16. ServerFormHandler
def ServerFormHandler(self):
    try:
        if len(self.soup.find_all('form', action=True))==0:
            return 1
        else :
            for form in self.soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank":
                    return -1
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
                else:
                    return 1
    except:
        return -1

# 17. InfoEmail
```

```
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?]", self.soap):
            return -1
        else:
            return 1
    except:
        return -1

# 18. AbnormalURL
def AbnormalURL(self):
    try:
        if self.response.text == self.whois_response:
            return 1
        else:
            return -1
    except:
        return -1

# 19. WebsiteForwarding
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
         return -1

# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 22. UsingPopupWindow
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
```

```
            return 1
        else:
            return -1
    except:
        return -1

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

# 24. AgeofDomain
def AgeofDomain(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass

        today  = date.today()
        age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
        if age >=6:
            return 1
        return -1
    except:
        return -1

# 25. DNSRecording
def DNSRecording(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass

        today  = date.today()
        age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
        if age >=6:
            return 1
        return -1
    except:
        return -1

# 26. WebsiteTraffic
def WebsiteTraffic(self):
    try:
```

```
            rank =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" +
url).read(), "xml").find("REACH")['RANK']
            if (int(rank) < 100000):
                return 1
            return 0
        except :
            return -1

    # 27. PageRank
    def PageRank(self):
        try:
            prank_checker_response =
requests.post("https://www.checkpagerank.net/index.php", {"name": self.domain})

            global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
            if global_rank > 0 and global_rank < 100000:
                return 1
            return -1
        except:
            return -1


    # 28. GoogleIndex
    def GoogleIndex(self):
        try:
            site = search(self.url, 5)
            if site:
                return 1
            else:
                return -1
        except:
            return 1

    # 29. LinksPointingToPage
    def LinksPointingToPage(self):
        try:
            number_of_links = len(re.findall(r"<a href=", self.response.text))
            if number_of_links == 0:
                return 1
            elif number_of_links <= 2:
                return 0
            else:
                return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt
|ow\.ly', url)
            ip_address = socket.gethostbyname(self.domain)
```

```
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|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|1
07\.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|21
3\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200
\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.1
56\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1

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

App.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <center> <h1> IBM Project Web Phishing Detection </h1> </center>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the safety of url.">
    <meta name="keywords" content="phishing url,phishing,cyber security,machine
learning,classifier,python">
    <meta name="author" content="Mahath P T">

    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">

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

```
<body>
 <center>  <img class="image image-contain" src="https://www.tworivercomputer.com/wp-content/uploads/sites/558/2020/10/bigstock-Data-Phishing-Hacker-Attack-t-319270852-1024x652.jpg" alt="MDN logo" /> </center>

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

         <br>
         <form action="/" method ="post">
            <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL" required="" />
            <label for="url" class="form__label">URL</label>
            <button class="button" role="button" >Check here</button>
         </form>

   </div>

   <div class="col-md" id="form2">

      <br>
      <h6 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h6>

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

   <!-- JavaScript -->
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
      crossorigin="anonymous"></script>
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
      integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
      crossorigin="anonymous"></script>


   <script>

         let x = '{{xx}}';
         let num = x*100;
```

```
        if (0<=x && x<0.50){
            num = 100-num;
        }
        let txtx = num.toString();
        if(x<=1 && x>=0.50){
            var label = "Website is "+txtx +"% safe to use...";
            document.getElementById("prediction").innerHTML = label;
            document.getElementById("button1").style.display="block";
        }
        else if (0<=x && x<0.50){
            var label = "Website is "+txtx +"% unsafe to use..."
            document.getElementById("prediction").innerHTML = label ;
            document.getElementById("button2").style.display="block";
        }

    </script>

</body>
<footer>
      <center> <p>© 2022 Mahath P T</p> </center>
    </footer>
</html>
```

App.css

```
*,
*::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
  font-size: 62,5%;
}
.image {
  width: 500px;
  height: 500px;
}
.image-contain {
  object-fit: contain;
  object-position: center;
}

.image-cover {
  object-fit: cover;
  object-position: center;
}
body {
  padding: 10% 5%;
  background: #0f2027;
  background: linear-gradient(to right,#2c5364, #203a43, ##55FFFF);
  justify-content: center;
  align-items: center;
  height: 100vh;
  color: #fff;
}

.form__label {
```

```
  font-family: 'Roboto', sans-serif;
  font-size: 1.2rem;
  margin-left: 2rem;
  margin-top: 0.7rem;
  display: block;
  transition: all 0.3s;
  transform: translateY(0rem);
}

.form__input {
  top: -24px;
  font-family: 'Roboto', sans-serif;
  color: #333;
  font-size: 1.2rem;
  padding: 1.5rem 2rem;
  border-radius: 0.2rem;
  background-color: rgb(255, 255, 255);
  border: none;
  width: 75%;
  display: block;
  border-bottom: 0.3rem solid transparent;
  transition: all 0.3s;
}

.form__input:placeholder-shown + .form__label {
  opacity: 0;
  visibility: hidden;
  -webkit-transform: translateY(+4rem);
  transform: translateY(+4rem);
}


.button {
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, #fff, #f8eedb);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI
Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
```

```
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
}

.button:active {
  background-color: #f3f4f6;
  box-shadow: -1px 2px 5px rgba(81,41,10,0.15),0px 1px 1px rgba(81,41,10,0.15);
  transform: translateY(0.125rem);
}

.button:focus {
  box-shadow: rgba(72, 35, 7, .46) 0 0 0 4px, -6px 8px 10px rgba(81,41,10,0.1), 0px 2px 2px
rgba(81,41,10,0.2);
}


.main-body{
  display: flex;
  flex-direction: row;
  width: 75%;
  justify-content:space-around;
}

.button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245, 174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI
Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}
```

```
.button2{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(252, 162, 162), #ee3737);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI
Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

.right {
  right: 0px;
  width: 300px;
}

@media (max-width: 576px) {
  .form {
    width: 100%;
  }
 }
.abc{
  width: 50%;
}
```

GITHUB

https://github.com/IBM-EPBL/IBM-Project-16461-1659614947

Demo Link

https://github.com/IBM-EPBL/IBM-Project-16461-1659614947/tree/main/Final%20Deliverables/Demo%20Video