PROJECT

WEB PHISHING DETECTION

DONE BY

TEAM ID : PNT2022TMID10245

M. DHINESH

R. JEFRIL ANGELAN

K.S. GOKULA KRISHNA

R. GUNASEKARAN

Project Report

1. **INTRODUCTION**
   1.1.Project Overview
   1.2.Purpose
2. **LITERATURE SURVEY**
   2.1.Existing problem
   2.2.References
   2.3.Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   3.1. Empathy Map Canvas
   3.2.Ideation & Brainstorming
   3.3.Proposed Solution
   3.4.Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   4.1. Functional requirement
   4.2.Non-Functional requirements
5. **PROJECT DESIGN**
   5.1. Data Flow Diagrams
   5.2. Solution & Technical Architecture
6. **PROJECT PLANNING & SCHEDULING**
   6.1. Sprint Planning & Estimation
   6.2.Sprint Delivery Schedule
   6.3.Reports from JIRA
7. **CODING & SOLUTIONING**
   7.1. Feature 1
   7.2. Feature 2
   7.3. Database Schema (if Applicable)
8. **TESTING**
   8.1. Test Cases
   8.2. User Acceptance Testing
9. **RESULTS**
   9.1. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE 13. APPENDIX**
    Source Code
    GitHub & Project Demo Link

1. INTRODUCTION

1.1. Project Overview

Phishing can be defined as impersonating a valid site to trick users by stealing their personal data comprising usernames, passwords, accounts numbers, national insurance numbers, etc. Phishing frauds might be the most widespread cybercrime used today. There are countless domains where phishing attack can occur like the online payment sector, webmail, financial institutions, file hosting or cloud storage and many others. The webmail and online payment sector was embattled by phishing more than in any other industry sector. Phishing can be done through email phishing scams and spear phishing hence user should be aware of the consequences and should not give their 100 percent trust on common security application. Machine Learning is one of the efficient techniques to detect phishing as it removes drawback of existing approach.

1.2. Purpose

The objectives which is the most vital thing in proposed project is to verify the validity of the website by capturing blacklisted URLs. To notify the user on blacklisted website through pop-up while they are trying to access and to notify the user on blacklisted website through email while they are trying to access. This proposed project will allow administrator to add blacklisted URL's in order to alert user during their inquiry.

The two scope of project, which is well known as user scope and system scope. User has some responsibility towards the system. The system includes a few standards and policies that requires to be obliged in order to comply the system. The user can be notified if blacklisted website is being accessed. The admin can capture the blacklisted URL's to alert user. The system involves features like capturing blacklisted website, viewing blacklisted website, displaying pop-up notification and also displaying email notification.

2. LITERATURE SURVEY

2.1. Existing problem

Couple of researchers have analysed the stats of malicious sites in some way. Our method picks up some of the important ideas from previous case studies. Ma, et al. [3,4] compared various batch-based learning algorithms used in classifying phishing sites and stated that a combination of host based and lexicalbased features outcome in the highest accuracy in classification. Besides, they are also compared with the performance of batch-based algorithms with the onlinebased algorithms which when utilizes complete features and noticed that onlinebased algorithms, especially Confidence-Weighted (CW), stand out performing batch-based algorithms. The attributes include the existence of the red flag keywords present in the website, attributes that are based on Google's Page Rank and Google's Web page quality guidelines. One cannot compare directly without access to the same websites and attributes.

2.2. References

[1] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs, "Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions," in Proceedings of the 28th international conference on Human factors in computing systems, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 373–382.

[2] B. Krebs, "HBGary Federal hacked by Anonymous," http: //krebsonsecurity.com/2011/02/hbgary-federal-hacked-by-anonymous/, 2011, accessed December 2011.

[3] B. Schneier, "Lockheed Martin hack linked to RSA's SecurID breach," http://www.schneier.com/blog/archives/2011/05/lockheed martin.html, 2011, accessed December 2011.

[4] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages," in NDSS '10, 2010.

[5] X. Dong, J. Clark, and J. Jacob, "Modelling user-phishing interaction," in Human System Interactions, 2008 Conference on, may 2008, pp. 627 –632.

[6] W. D. Yu, S. Nargundkar, and N. Tiruthani, "A phishing vulnerability analysis of web based systems," in Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC 2008). Marrakech, Morocco: IEEE, July 2008, pp. 326–331.

[7] Anti-Phishing Working Group (APWG), "Phishing activity trends report — second half 2010," http://apwg.org/reports/apwg report h2 2010. pdf, 2010, accessed December 2011.

[8] Anti-Phishing Working Group (APWG), "Phishing activity trends report — first half 2011," http://apwg.org/reports/apwg trends report h1 2011.pdf, 2011, accessed December 2011.

[9] Anti-Phishing Working Group (APWG), "Phishing activity trends report — second half 2011," http://apwg.org/reports/apwg trends report h2 2011.pdf, 2011, accessed July 2012.

[10] B. Schneier, "Details of the RSA hack," http://www.schneier.com/blog/ archives/2011/08/details of the.html, 2011, accessed December 2011

2.3. Problem Statement Definition

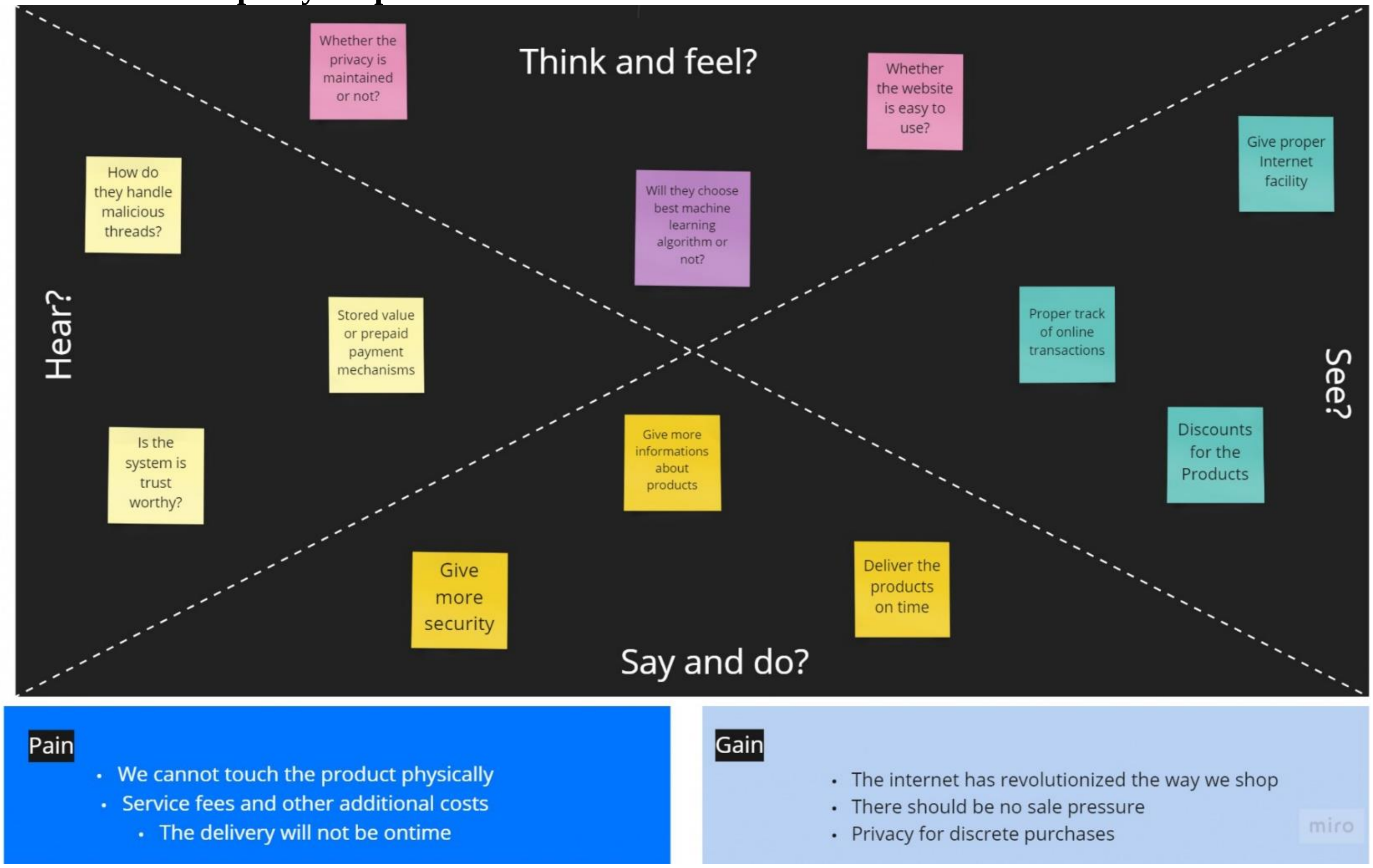
Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. Furthermore, page content inspection has been used by some strategies to overcome the false negative problems and complement the vulnerabilities of the stale lists.

Moreover, page content inspection algorithms each have different approach to phishing website detection with varying degrees of accuracy. Therefore, ensemble can be seen to be a better solution as it can combine the similarity in accuracy and different error-detection rate properties in selected algorithms. Therefore, this study will address a couple of research:

1. How to process raw dataset for phishing detection?
2. How to increase detection rate in phishing websites algorithms?
3. How to reduce false negative rate in phishing websites algorithm?
4. What are the best compositions of classifiers that can give a good detection rate of phishing website?

3. IDEATION & PROPOSED SOLUTION

3.1. Empathy Map Canvas


Think and feel?
Whether the privacy is maintained or not?
Whether the website is easy to use?
Will they choose best machine learning algorithm or not?
Hear?
How do they handle malicious threads?
Stored value or prepaid payment mechanisms
Is the system is trust worthy?
See?
Give proper Internet facility
Proper track of online transactions
Discounts for the Products
Say and do?
Give more informations about products
Give more security
Deliver the products on time
miro

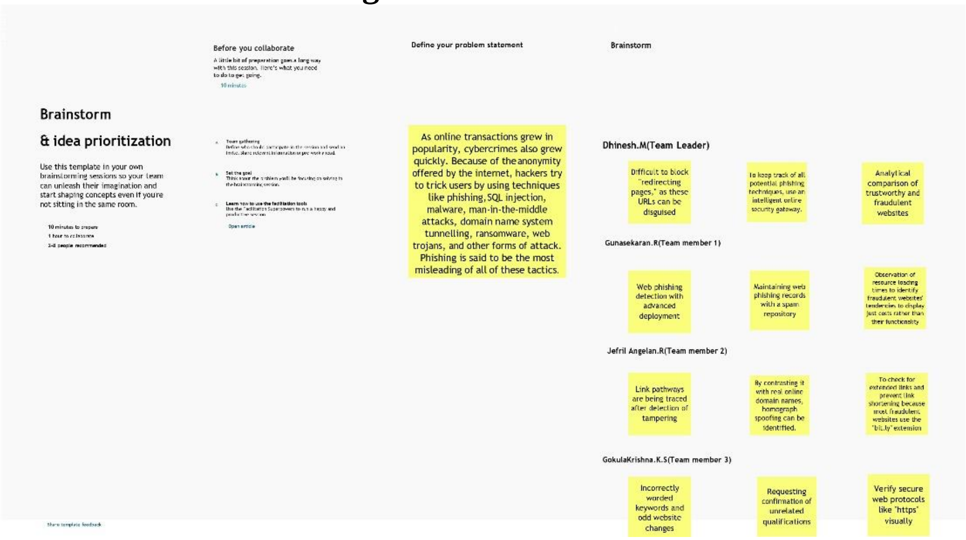
Pain

- We cannot touch the product physically
- Service fees and other additional costs
- The delivery will not be ontime

Gain

- The internet has revolutionized the way we shop
- There should be no sale pressure
- Privacy for discrete purchases

3.2. Ideation & Brainstorming



Brainstorm

& idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if youre not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
3-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
Open article

Define your problem statement

As online transactions grew in popularity, cybercrimes also grew quickly. Because of the anonymity offered by the internet, hackers try to trick users by using techniques like phishing, SQL injection, malware, man-in-the-middle attacks, domain name system tunnelling, ransomware, web trojans, and other forms of attack. Phishing is said to be the most misleading of all of these tactics.

Brainstorm

Dhinesh.M(Team Leader)

- Difficult to block "redirecting pages," as these URLs can be disguised
- To keep track of all potential phishing techniques, use an intelligent online security gateway.
- Analytical comparison of trustworthy and fraudulent websites

Gunasekaran.R(Team member 1)

- Web phishing detection with advanced deployment
- Maintaining web phishing records with a spam repository
- Observation of resource loading times to identify fraudulent websites' tendencies to display just costs rather than their functionality

Jefril Angelan.R(Team member 2)

- Link pathways are being traced after detection of tampering
- By contrasting it with real online domain names, homograph spoofing can be identified.
- To check for extended links and prevent link shortening because most fraudulent websites use the "bit.ly" extension

GokulaKrishna.K.S(Team member 3)

- Incorrectly worded keywords and odd website changes
- Requesting confirmation of unrelated qualifications
- Verify secure web protocols like "https" visually

Share template feedback

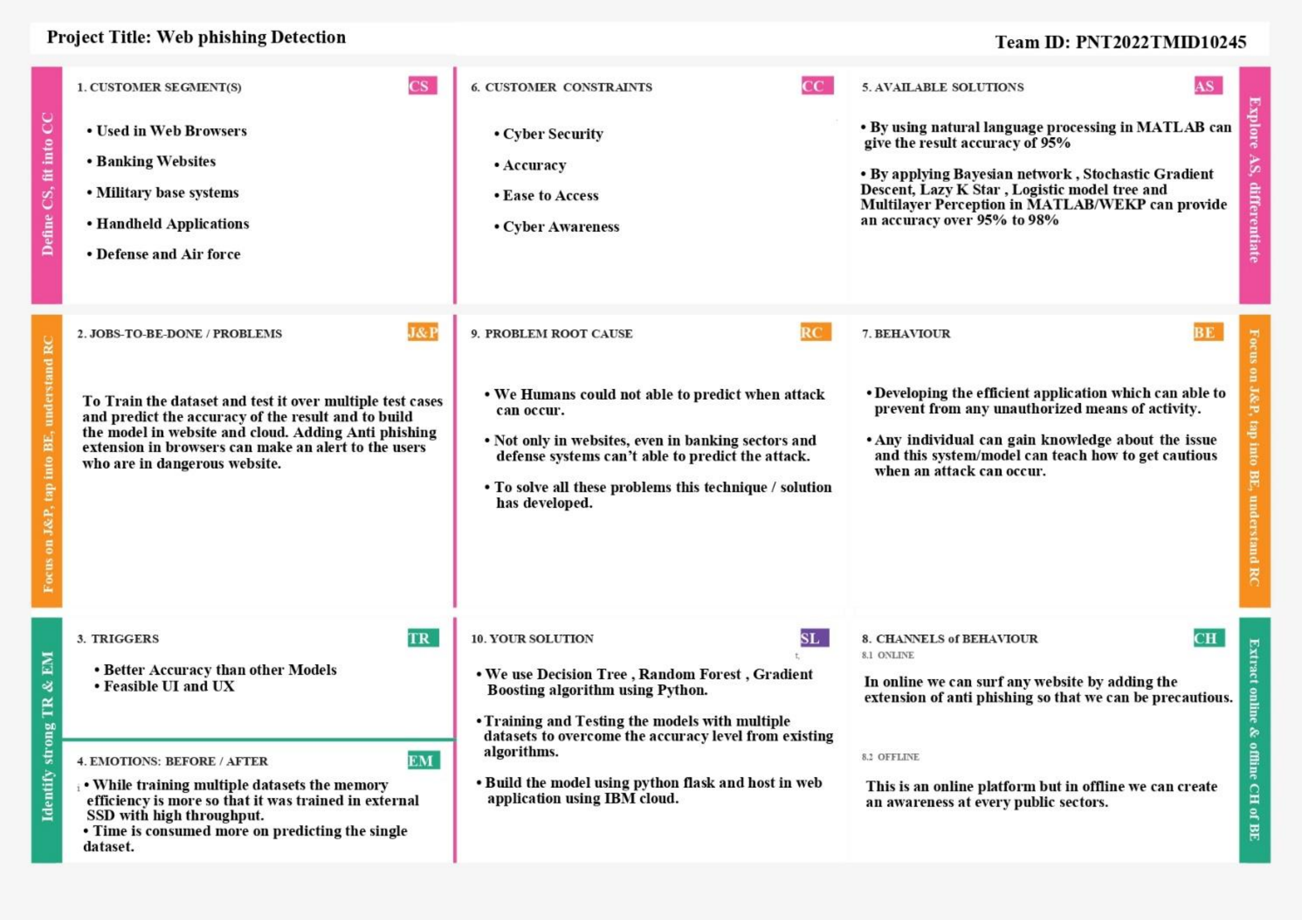
3.3. Proposed Solution

S. No.	Parameter	Description
1.	**Problem Statement (Problem to be solved)**	Phishing sites are malicious website that aim to steal user's personal data. Spotting these phishing website is typically a challenging task because phishing is mainly a semantic-based attack that mainly focused on software vulnerability etc.

2.	**Idea/Solution description**	Our product server as a browser extension and it scrapes the website URL and runs it through our ML model. If the model detects it as a phishing website, the extension notifies the user.

3.	**Novelty / Uniqueness**	The browser extension factors is not used in any previous works. The user does not have to think twice before using a website,our extention will take care of the classifying work.
4.	**SocialImpact/ CustomerSatisfaction**	Reduce the amount of information stolen by phishing sites and also increase customer satisfaction as they would be reassured when using legitimate website.
5.	**BusinessModel (Revenue Model)**	We propose a two tier system namely a FREE and PREMIUM tire. The FREE tier would include ads and the PREMIUM tier is a recurring subscription either monthly or annually.
6.	**Scalability of the Solution**	Since this is a browser extension which would be published in Chrome Marketplace,it can be accessed and used by everyone across the world.

3.4. Problem Solution fit



Project Title: Web phishing Detection

Team ID: PNT2022TMID10245

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Used in Web Browsers
- Banking Websites
- Military base systems
- Handheld Applications
- Defense and Air force

6. CUSTOMER CONSTRAINTS CC

- Cyber Security
- Accuracy
- Ease to Access
- Cyber Awareness

5. AVAILABLE SOLUTIONS AS

- By using natural language processing in MATLAB can give the result accuracy of 95%
- By applying Bayesian network , Stochastic Gradient Descent, Lazy K Star , Logistic model tree and Multilayer Perception in MATLAB/WEKP can provide an accuracy over 95% to 98%

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

To Train the dataset and test it over multiple test cases and predict the accuracy of the result and to build the model in website and cloud. Adding Anti phishing extension in browsers can make an alert to the users who are in dangerous website.

9. PROBLEM ROOT CAUSE RC

- We Humans could not able to predict when attack can occur.
- Not only in websites, even in banking sectors and defense systems can't able to predict the attack.
- To solve all these problems this technique / solution has developed.

7. BEHAVIOUR BE

- Developing the efficient application which can able to prevent from any unauthorized means of activity.
- Any individual can gain knowledge about the issue and this system/model can teach how to get cautious when an attack can occur.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- Better Accuracy than other Models
- Feasible UI and UX

4. EMOTIONS: BEFORE / AFTER EM

- While training multiple datasets the memory efficiency is more so that it was trained in external SSD with high throughput.
- Time is consumed more on predicting the single dataset.

10. YOUR SOLUTION SL

- We use Decision Tree , Random Forest , Gradient Boosting algorithm using Python.
- Training and Testing the models with multiple datasets to overcome the accuracy level from existing algorithms.
- Build the model using python flask and host in web application using IBM cloud.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

In online we can surf any website by adding the extension of anti phishing so that we can be precautious.

8.2 OFFLINE

This is an online platform but in offline we can create an awareness at every public sectors.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1. Functional requirement

FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Learning & Detection	The samples and the topological structure of the machine learning TensorFlow is built. The submitted URLs are tested against thesamples in the database to perform classification.

FR-2	Testing & Alert	URLs passed through the system are recorded in a database, thus each URL submitted by the user is tested to check or duplicate. If a phishing website is detected the popup message will alert the user. Give information about the malicious website with accurate result.
FR-3	Deep Learning	The phishing detection process could be doneusing the Recurrent Neural Network.The website could bedetected.
FR-4	Hardware Requirements	2GB RAM(minimum) 100GB HDD(minimum) Intel i3 quad core 1.66GHz processor(minimum) Internet Connectivity
FR-5	Software Requirements	Windows 7 or higher Python 3.6.0 or higher Visual Studio Code Flask(python platform) HTML Dataset consisting of Phishing websites and their features. Required plugins and libraries Jupiter notebook
FR-6	Other requirements	IBM cloud login Chrome extension features

4.2. Non-Functional requirements

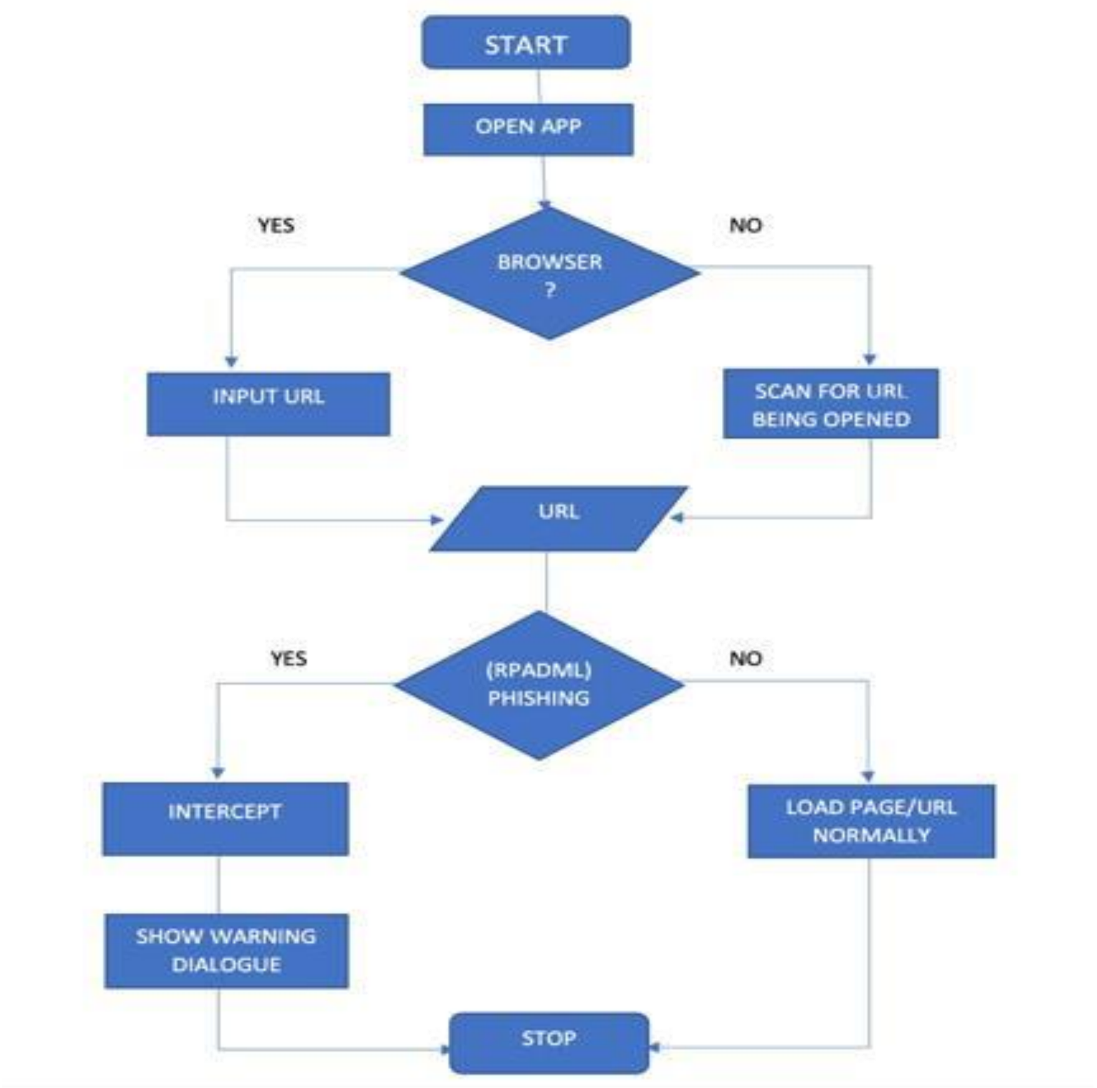
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	This system is really used as it can able to detect phishing websites. By detecting malicious websites, our personal and professional data are confidential, secure, and accessible.

NFR-2	**Security**	Phishers spoof legitimate emails so that the victim trusts them. They send out massive numbers of fraudulent emails in order to catch a smallpercentage of recipients off guard. They create a sense of urgency so that the victim does not think twice before clicking the link or downloading the attachment. Lack of security awareness among employeesis also one of the major reasons for the success of phishing. Organizations should be aware ofhow the benefits and purpose of security awareness training can secure their employees from falling victim to phishing attacks.
NFR-3	**Reliability**	The performance of the system wouldbe accurate. Probability of giving false information is verylow. As the system is working based on the deep learning algorithm, it would easily predict and give the perfect information.
NFR-4	**Performance**	The effectiveness of these methods relies on feature collection, training data, and classification algorithms and giving alerts when phished websites are detected. It must be processed and executed within a fraction of a second using the deep learning algorithm
NFR-5	**Availability**	The availability of the solution is effective and it should be helpful in a great way to prevent our personal data to be exposed.

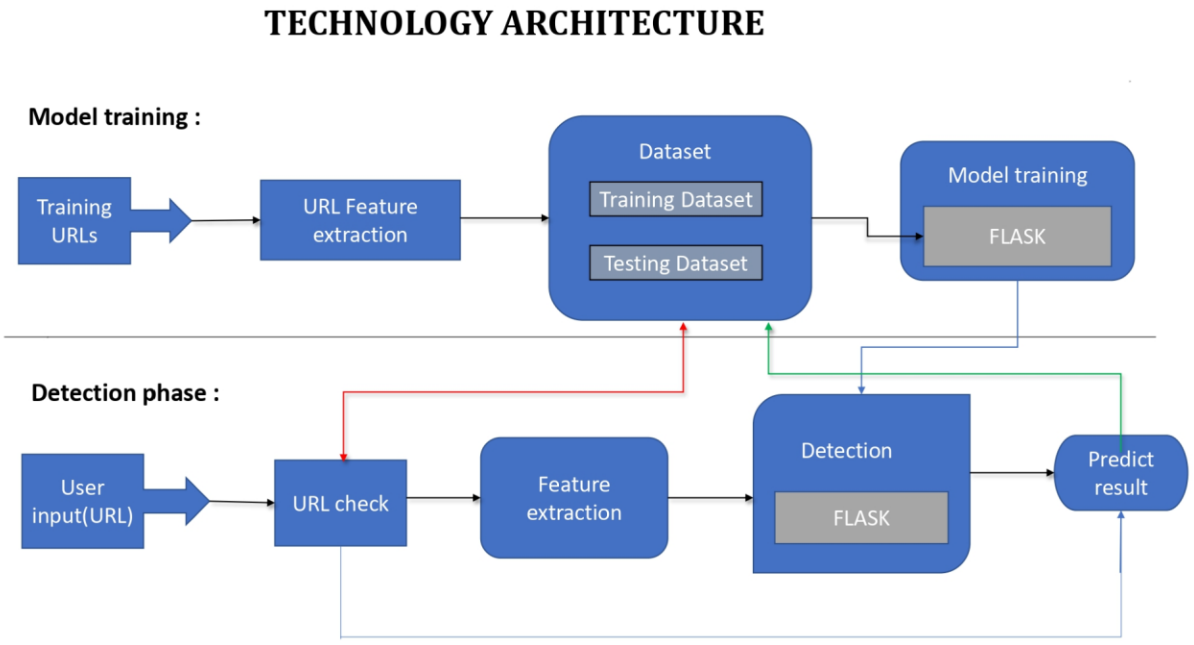
NFR-6	**Scalability**	This solution is scalable enough to fit theSecurity issues by constructing the best website. The cost of establishing the website and maintaining all the programs may be high . It is acceptable to fit them over any place and any resources.

5. PROJECT DESIGN

5.1. Data Flow Diagrams



5.2. Solution & Technical Architecture



6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	URL detector	USN-1	URL is the first thing to analyze a website to decide whether it is a phishing or not	10	High	M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran

Sprint-1		USN-2	Some of URL-Based Features are ● Digit count in the URL ● Total length of URL ● Checking whether the URL is typo-squatted or not ● Checking whether it includes a legitimate brand name or not ● Number of subdomains in URL ● TLD is one of the commonly used one	10	High	M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran
Sprint-2	Domain detection	USN-3	The purpose of Phishing Domain Detection is detecting phishing domain names. Therefore, passive queries related to the domain name, which we want to classify as phishing or not, provide useful information to us.	10	High	M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran

Sprint-2		USN-4	Some useful Domain-Based Features are ● Its domain name or its IP address in blacklists of wellknown reputation services? ● How many days passed since the domain was registered? ● Is the registrant name hidden?	10	High	M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran
Sprint-3	Page based features and Content based features	USN-5	Page-Based Features are using information aboutpages which are calculated reputation ranking services. Obtaining these types of features requires active scan to target domain. Page contents are processed for us to detect whether target domain is used for phishing or not	10	High	M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran

Sprint-3			● Global pagerank ● Country pagerank ● Position at the Alexa top 1 million site Some processed information about pages are ● Page titles ● Meta tags ● Hidden text ● Text in the body ● Images etc.			M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran
Sprint-4	Detection process	USN-6	Detecting Phishing Domains is a classification problem, so it means we need labeled data which has samples as phish domains and legitimate domains in the training phase	20		M.Dhinesh R.Jefril Angelan K.S.Gokula Krishna R.Gunasekaran

Project Tracker, Velocity & Burndown Chart:

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

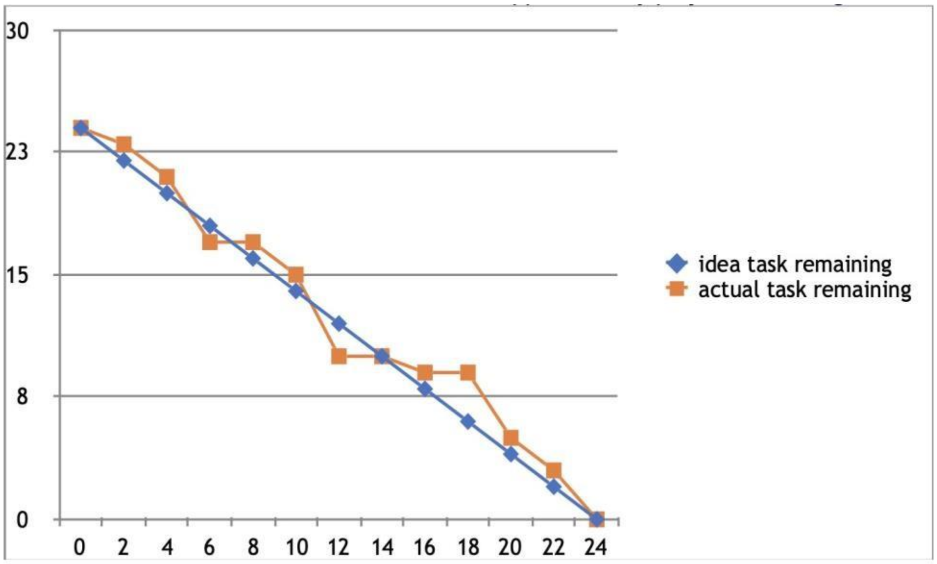
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

Burndown Chart:

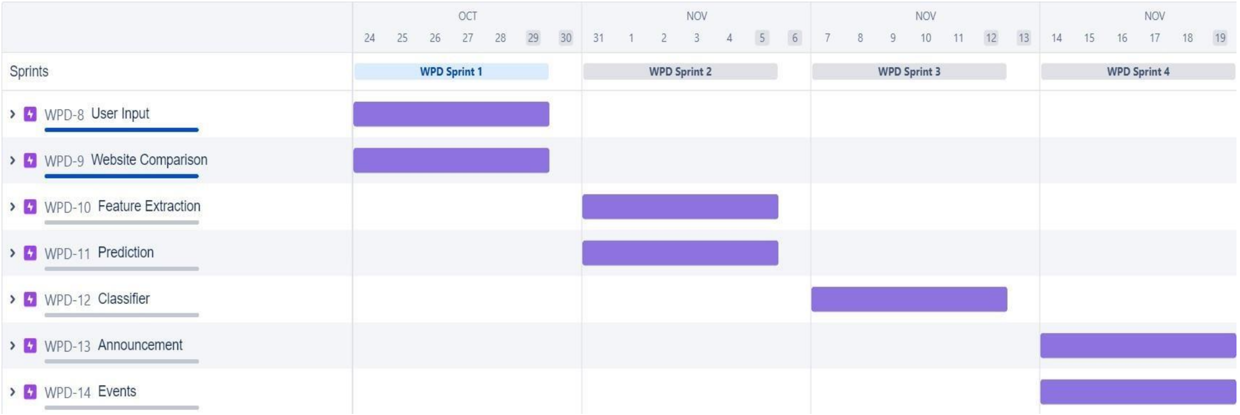
A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.2. Sprint Delivery Schedule

Sprint	Sprint Topic	Start Date	Expected Delivery
Sprint 1	URL detector	24-10-2022	29-10-2022
Sprint 2	Domain detection	31-10-2022	05-11-2022
Sprint 3	Page based features and content based features	07-11-2022	12-11-2022
Sprint 4	Detection process	14-11-2022	19-11-2022

6.3. Reports from JIRA



7. CODING & SOLUTIONING

7.1. Feature 1

This feature is used to import required libraries to load the model from the .pkl file which was builded in the model building phase.

Coding :

```
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("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 = "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=2002)
```

7.2. Feature 2

This feature helps in providing easy UI to the user using the web interface.
Coding :

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <center> <h1> IBM Project Based Learning </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="Balajee A V">

   <!-- BootStrap -->
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/
4.5.0/css/bootstrap.min.css"
      integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1d
KGj7Sk" crossorigin="anonymous">

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

<body>
 <center> <img class="image image-contain" src="https://cdn.activestate.com/
wp-content/uploads/2021/02/phishing-detection-with-Python.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+OGpamo-
FVy38MVBnE+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+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMf
ooAo"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
```

```
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR
0JKI"
        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>
</html>
```

8. TESTING

8.1. Test Cases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To ExecUte	Test Data	Expected Result	Actual Result	Status	Comments	TC forAutomation (Y/ N)	BUG ID	Executed By
LoginPage_TC_OO1	UI	Home Page	Verify the UIelements 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://www.google.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		M.Dhinesh

LoginPag e_TC_OO2	Functional	Home page	Verify whether the link is legitimat e 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://www.youtube.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		R.Jefril Angelan
LoginPag e_TC_OO3	Functional	Home Page	Verify user is able to access the legitimat e 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 legitimat e or be cautious if it is not legitimat e.	http://ssalescript.info/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		K.S.Gokul Krishna

LoginPage_TC_OO4	Functional	Home Page	Testing thewebsite with multiple URLs		1. Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL totest 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	https://www.delgets.com/	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		R.Gunasekaran

8.2. User Acceptance Testing

1. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0

Won't Fix	0	0	2	1	3
Totals	23	9	12	25	60

1. Test Case Analysis

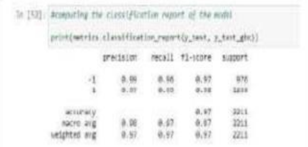
This report shows the number of test cases that have passed, failed, and untested

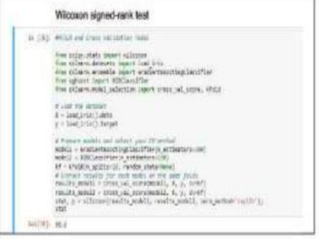
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

9. RESULTS

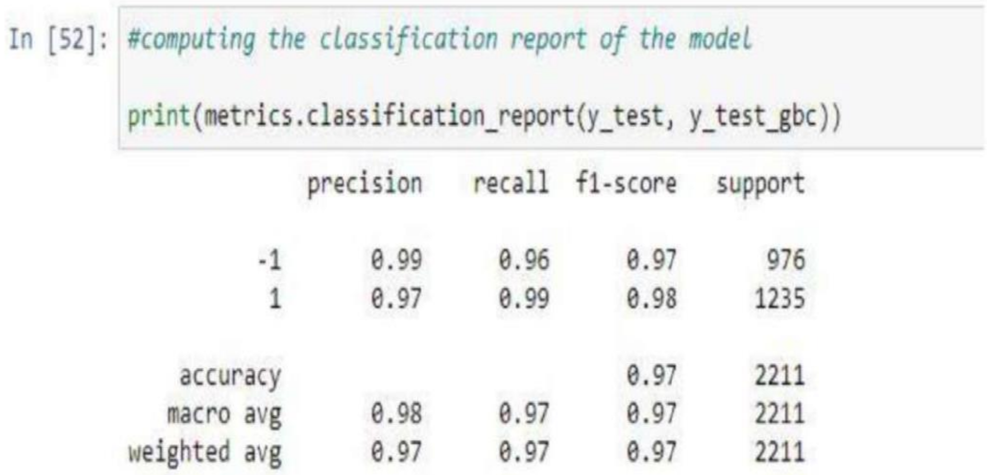
9.1. Performance Metrics

S.No.	Parameter	Values	Screenshot

1.	Metrics	**Classification Model:**
		Gradient Boosting Classification
		Accuray Score- 97.4%
			
			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

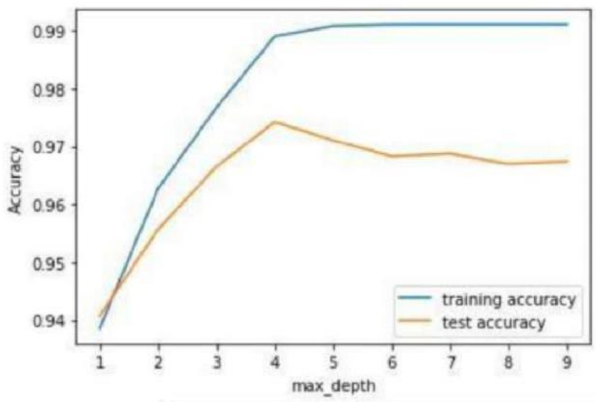
2.	Tune the Model	Hyperparameter Tuning - 97%
		Validation Method – KFOLD & Cross Validation Method
			
			Wilcoxon signed-rank test
			In [9]: #KFOLD and Cross Validation Model
			from scipy.stats import wilcoxon
			from sklearn.datasets import load_iris
			from sklearn.ensemble import GradientBoostingClassifier
			from xgboost import XGBClassifier
			from sklearn.model_selection import cross_val_score, KFold
			# Load the dataset
			X = load_iris().data
			y = load_iris().target
			# Prepare models and select your CV method
			model1 = GradientBoostingClassifier(n_estimators=100)
			model2 = XGBClassifier(n_estimators=100)
			kf = KFold(n_splits=10, random_state=None)
			# Extract results for each model on the same folds
			results_model1 = cross_val_score(model1, X, y, cv=kf)
			results_model2 = cross_val_score(model2, X, y, cv=kf)
			stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
			stat
			Out[9]: 95.0

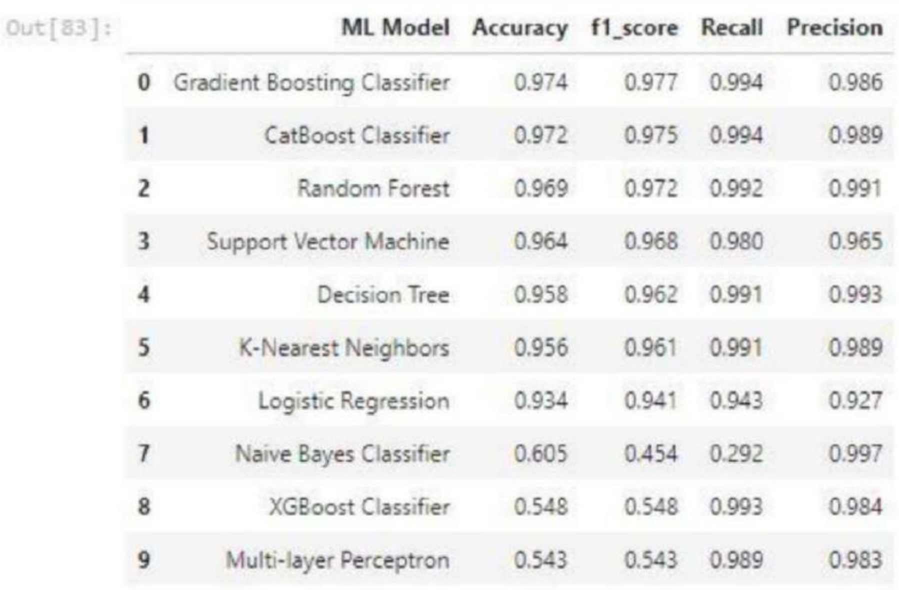
1. METRICS:
CLASSIFICATION REPORT:



```
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
```

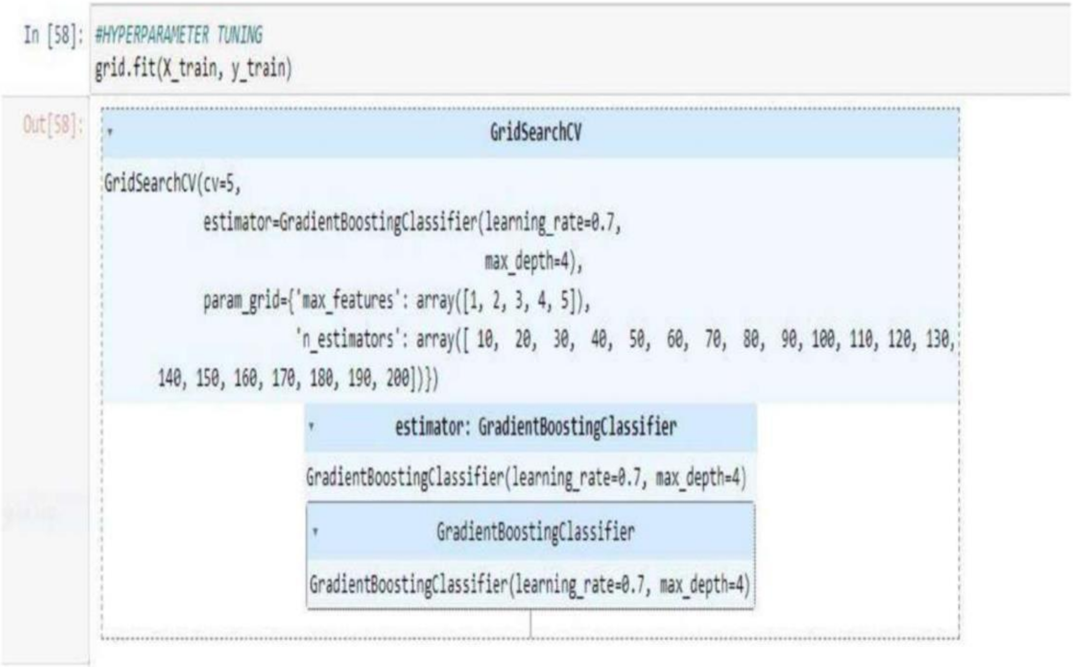
PERFORMANCE :




Out[83]:

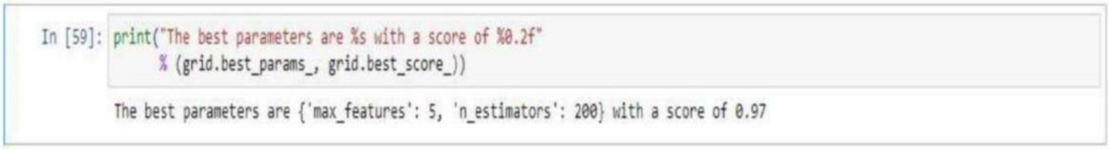
	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	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

2. TUNE THE MODEL – HYPERPARAMETER TUNING



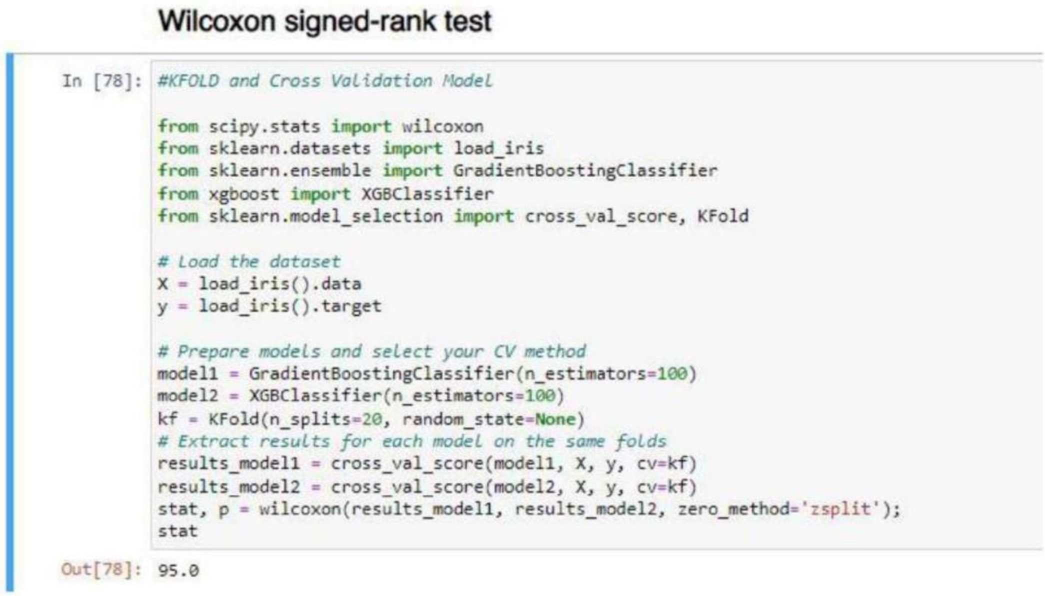
```
In [58]: #HYPERPARAMETER TUNING
         grid.fit(X_train, y_train)
```

```
Out[58]:
GridSearchCV

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

estimator: GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

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



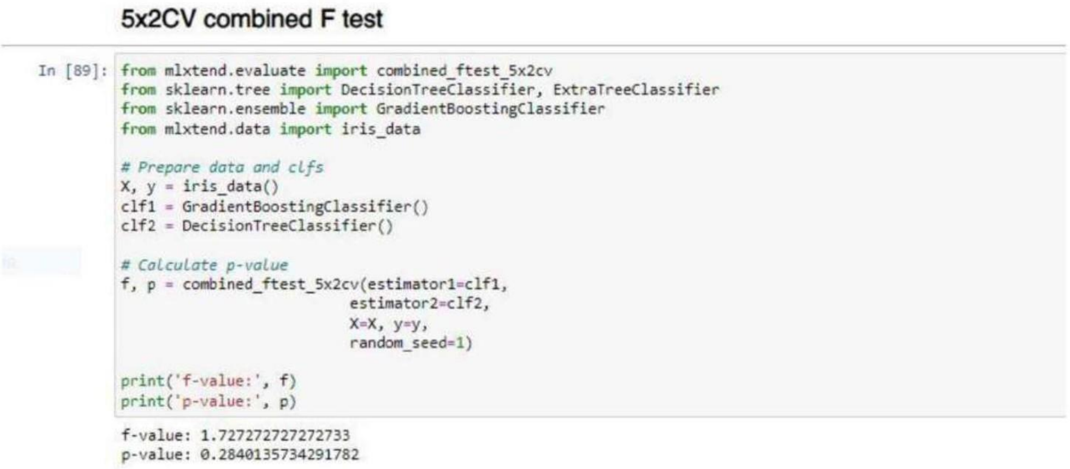
```
In [59]: print("The best parameters are %s with a score of %0.2f"
               % (grid.best_params_, grid.best_score_))

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

VALIDATION METHODS: KFOLD & Cross Folding



Wilcoxon signed-rank test

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

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

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

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

Out[78]: 95.0
```



5x2CV combined F test

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

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

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

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

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

10. ADVANTAGES & 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

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to tackle a wide assortment of classification issues, the procedure of finding the ideal structure is very difficult, and much of the time, this structure is controlled by experimentation.Our model takes care of this issue via computerizing the way toward organizing a neural system conspire; hence, on the off chance that we construct an enemy of phishing model and for any reasons we have to refresh it, at that point our model will encourage this procedure, that is, since our model will mechanize the organizing procedure and will request scarcely any client defined parameters.

12. FUTURE SCOPE

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique.In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features,Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

13. APPENDIX

A mechanism to detect phishing websites. Our methodology uses not just traditional URL based or content based rules but rather employs the machine learning technique to identify not so obvious patterns and relations in the data. We have used features from various domain spanning from URL to HTML tags of the webpage, from embedded URLs to favicon, and databases like WHOIS, Alexa, Pagerank, etc. to check the traffic and status of the website. We were able to obtain an accuracy of more than 96%, recall greater than 96% with a False Positive Rate of less than 5%, thus classifying most websites correctly and proving the effectiveness of the machine learning based technique to attack the problem of phishing websites. We provided the output as a user-friendly web platform which can further be extended to a browser extension to provide safe and healthy online space to the users.

Source Code:

```
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\.i
    nk|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.  AnchorURLdef
    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.  AbnormalURLdef
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.  AgeofDomaindef
    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.checkpager-
ank.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|107\.151\.148\.109|119\.28\.52\.61|
54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'
                                          '118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|
                 104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|
43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'
'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|
199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|
208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'
                                       '34\.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\.156\.19|37\.157\.192\.102|
204\.11\.56\.48|110\.34\.231\.42',
   ip_address)          if url_match:
   return -1           elif ip_match:
```

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


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

GitHub link https://github.com/IBM-EPBL/IBM-Project-11788-1659345926