

# Project Report

**PROJECT NAME: WEB PHISHING DETECTION**

***TEAM ID: PNT2022TMID49434***

Team Lead : Sanjana S

Team Member 1: Rohini KS

Team Member 2: Sabitha M

Team Member 3: Angala Bharathi C

Team Member 4: Roshini G

## **1. INTRODUCTION**

1.1 Project Objectives

1.2 Project Purpose

1.3 Project Flow

## **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

5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

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

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

## **8. TESTING**

## **9. RESULTS**

9.1 Performance Metrics

## **10. ADVANTAGES & DISADVANTAGES**

## **11. CONCLUSION & FUTURE SCOPE**

## **12. APPENDIX**

Source Code

GitHub & Project Demo Link

# 1. INTRODUCTION

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.

## 1.1 Project Objectives:

### By the end of this project

We'll be able to understand the problem to classify if it is a regression or a classification kind of problem. We will be able to know how to pre-process/clean the data using different data pre-processing techniques. Applying different algorithms according to the dataset. We will be able to know how to find the accuracy of the model. We will be able to build web applications using the Flask framework.

## 1.2 Project Purpose

The main purpose of the project is to detect the fake or phishing websites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials. We are using machine learning algorithms to safeguard the sensitive data and to detect the phishing websites who are trying to gain access on sensitive data.

## 1.3 Project Flow

Find below the project flow to be followed while developing the project - Download the dataset. - Preprocess or clean the data. -Analyze the pre-processed data. -Train the machine with preprocessed data using an appropriate machine learning algorithm. -Save the model and its dependencies. -Build a Web application using a flask that integrates with the model built.

Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams

## 2. Literature Survey

### 2.1 Existing problem

#### **A.Protecting user against phishing using Anti- phishing:**

AntiPhish is used to avoid users from using fraudulent web sites which in turn may lead to phishing attack. Here, AntiPhish traces the sensitive information to be filled by the user and alerts the user whenever he/she is attempting to share his/her information to a untrusted web site. The much effective elucidation for this is cultivating the users to approach only for trusted websites.

However, this approach is unrealistic. Anyhow, the user may get tricked. Hence, it becomes mandatory for the associates to present such explanations to overcome the problem of phishing. Widely accepted alternatives are based on the creepy websites for the identification of “clones” and maintenance of records of phishing websites which are in hit list.

#### **B.Learning to Detect Phishing Emails:**

An alternative for detecting these attacks is a relevant process of reliability of machine on a trait intended for the reflection of the besieged deception of user by means of electronic communication. This approach can be used in the detection of phishing websites, or the text messages sent through emails that are used for trapping the victims. Approximately, 800 phishing mails and 7,000 non- phishing mails are traced till date and are detected accurately over 95% of them along with the categorization on the basis of 0.09% of the genuine emails. We can just wrap up with the methods for identifying the deception, along with the progressing nature of attacks.

#### **C.Phishing detection system for e-banking using fuzzy data mining:**

Phishing websites, mainly used for e-banking services, are very complex and dynamic to be identified and classified. Due to the involvement of various ambiguities in the detection, certain crucial data mining techniques may prove an effective means in keeping the e-commerce websites safe since it deals with considering various quality factors rather than exact values.

In this paper, an effective approach to overcome the “fuzziness” in the e-banking phishing website assessment is used an intelligent resilient and effective model for detecting e-banking phishing websites is put forth. The applied model is based on fuzzy logics along with data mining algorithms to consider various effective factors of the e-banking phishing website.

#### **D. Collaborative Detection of Fast Flux Phishing Domains:**

Here, two approaches are defined to find correlation of evidences from multiple servers of DNS and multiple suspects of FF domain. Real life examples can be used to prove that our correlation approaches expedite the detection of the FF domain, which are based on an analytical model which can quantify various DNS queries that are required to verify a FF

domain.

It also shows implementation of correlation schemes on a huge level by using a distributed model, that is more scalable as compared to a centralized one, is publish N subscribe correlation model known as LARSID. In deduction, it is quite difficult to detect the FF domains in a accurate and timely manner, as the screen of proxies is used to shield the FF Mother ship. A theoretical approach is used to analyze the problem of FF detection by calculating the number of DNS queries required to get back a certain amount of unique IP addresses.

#### **E. A Prior-based Transfer Learning Method for the Phishing Detection:**

A logistic regression is the root of a priority based transferrable learning method, which is presented here for our classifier of statistical machine learning. It is used for the detection of the phishing websites depending on our selected characteristics of the URLs. Due to the divergence in the allocation of the features in the distinct phishing areas, multiple models are proposed for different regions.

## **2.2 References**

- [1] Gunter Ollmann, "The Phishing Guide Understanding & Preventing Phishing Attacks", IBM Internet Security Systems, 2007.
- [2] <https://resources.infosecinstitute.com/category/enterprise/phishing/the-phishing-landscape/phishing-data-attack-statistics/#gref>
- [3] Mahmoud Khonji, Youssef Iraqi, "Phishing Detection: A Literature Survey IEEE, and Andrew Jones, 2013
- [4] Mohammad R., Thabtah F. McCluskey L., (2015) Phishing websites dataset. Available: <https://archive.ics.uci.edu/ml/datasets/Phishing+Websites> Accessed January 2016
- [5] <http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/>
- [6] <http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learning/>
- [7] <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
- [8] [www.alexa.com](http://www.alexa.com)
- [9] [www.phishtank.com](http://www.phishtank.com)

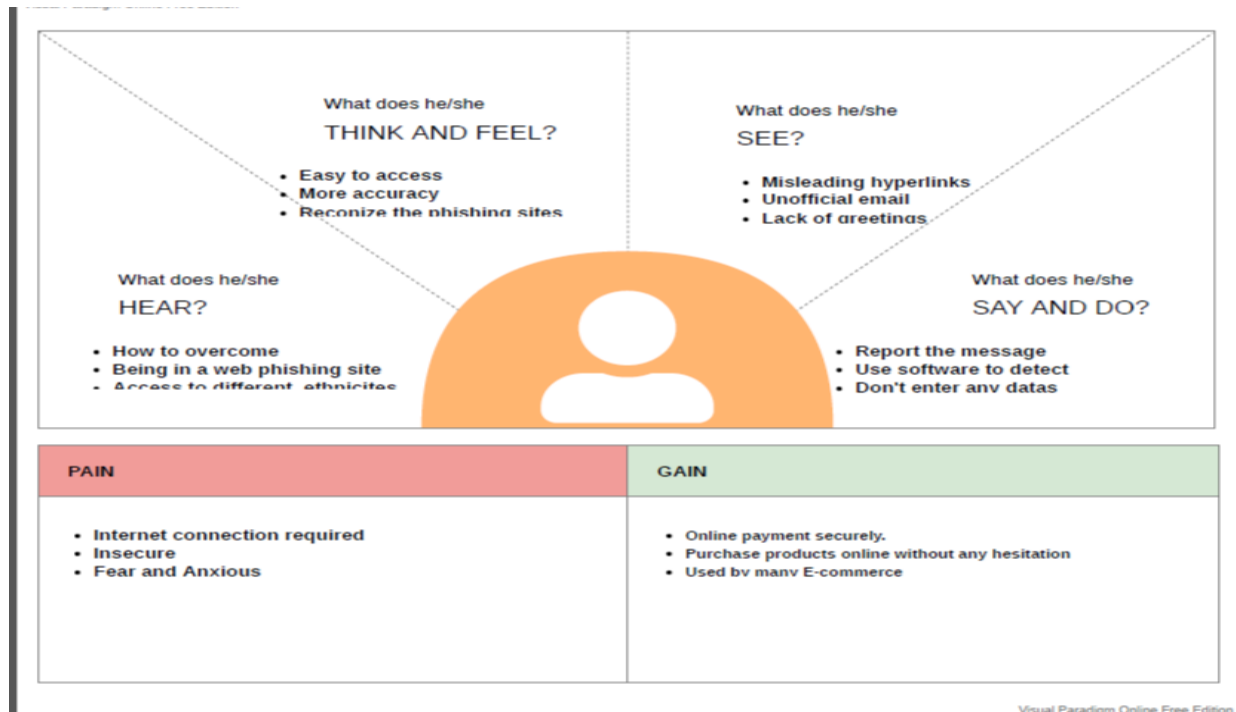
## **2.3 Problem Statement Definition**

This paper aims to enhance detection method to detect phishing websites using machine learning technology. We achieved 97.37% detection accuracy using random forest algorithm with lowest false positive rate. Also result shows that classifiers give better performance when we used more data as training data.

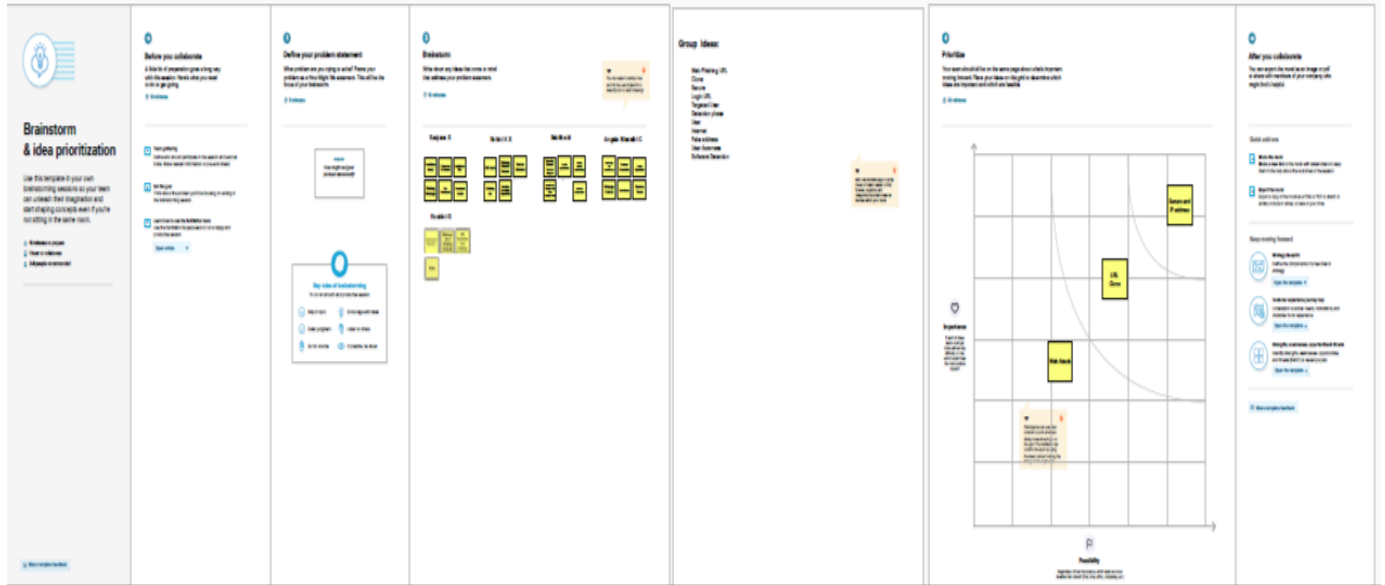
In future hybrid technology will be implemented to detect phishing websites more accurately, for which random forest algorithm of machine learning technology and blacklist method will be used.

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas



#### 3.2 Ideation & Brainstorming



### 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.	Social Impact / Customer Satisfaction	Reduce the amount of information stolen by phishing sites and also increase customer satisfaction as they would be reassured when using legitimate website.

5.	Business Model (Revenue Model)	We propose a two tier system namely a FREE and PREMIUM tier. 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: PNT2022TMID49434		
Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> <ul style="list-style-type: none"> <li>Used in Web Browsers</li> <li>Banking Websites</li> <li>Military base systems</li> <li>Handheld Applications</li> <li>Defense and Air force</li> </ul>	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> <ul style="list-style-type: none"> <li>Cyber Security</li> <li>Accuracy</li> <li>Ease to Access</li> <li>Cyber Awareness</li> </ul>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> <ul style="list-style-type: none"> <li>By using natural language processing in MATLAB can give the result accuracy of 95%</li> <li>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%</li> </ul>	Explore AS, differentiate	
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> <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.</p>	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> <ul style="list-style-type: none"> <li>We Humans could not able to predict when attack can occur.</li> <li>Not only in websites, even in banking sectors and defense systems can't able to predict the attack.</li> <li>To solve all these problems this technique / solution has developed.</li> </ul>	<b>7. BEHAVIOUR</b> <span>BE</span> <ul style="list-style-type: none"> <li>Developing the efficient application which can able to prevent from any unauthorized means of activity.</li> <li>Any individual can gain knowledge about the issue and this system/model can teach how to get cautious when an attack can occur.</li> </ul>	Focus on J&P, fit into RC, understand RC	
Identify strong TR & EM	<b>3. TRIGGERS</b> <span>TR</span> <ul style="list-style-type: none"> <li>Better Accuracy than other Models</li> <li>Feasible UI and UX</li> </ul>	<b>10. YOUR SOLUTION</b> <span>SL</span> <ul style="list-style-type: none"> <li>We use Decision Tree, Random Forest, Gradient Boosting algorithm using Python.</li> <li>Training and Testing the models with multiple datasets to overcome the accuracy level from existing algorithms.</li> <li>Build the model using python flask and host in web application using IBM cloud.</li> </ul>	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <p><b>8.1 ONLINE</b> In online we can surf any website by adding the extension of anti phishing so that we can be precautionous.</p> <p><b>8.2 OFFLINE</b> This is an online platform but in offline we can create an awareness at every public sectors.</p>	Extend online & offline CH of BE	
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> <ul style="list-style-type: none"> <li>While training multiple datasets the memory efficiency is more so that it was trained in external SSD with high throughput.</li> <li>Time is consumed more on predicting the single dataset.</li> </ul>				



## 4.REQUIREMENT ANALYSIS

### 4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement(Epic)	Sub Requirement(Story/Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN Registration through websites.
FR-2	User Confirmation	Confirmation via Email Conformation via OTP
FR-3	User Authentication	Conformation for email confirmation for passwords
FR-4	User Security	Strong Passwords,Two facxtor authentication,updating device management
FR-5	User Performance	Usage of legitimate websites,optimize network traffic

### 4.2 Non-Functional Requirements:

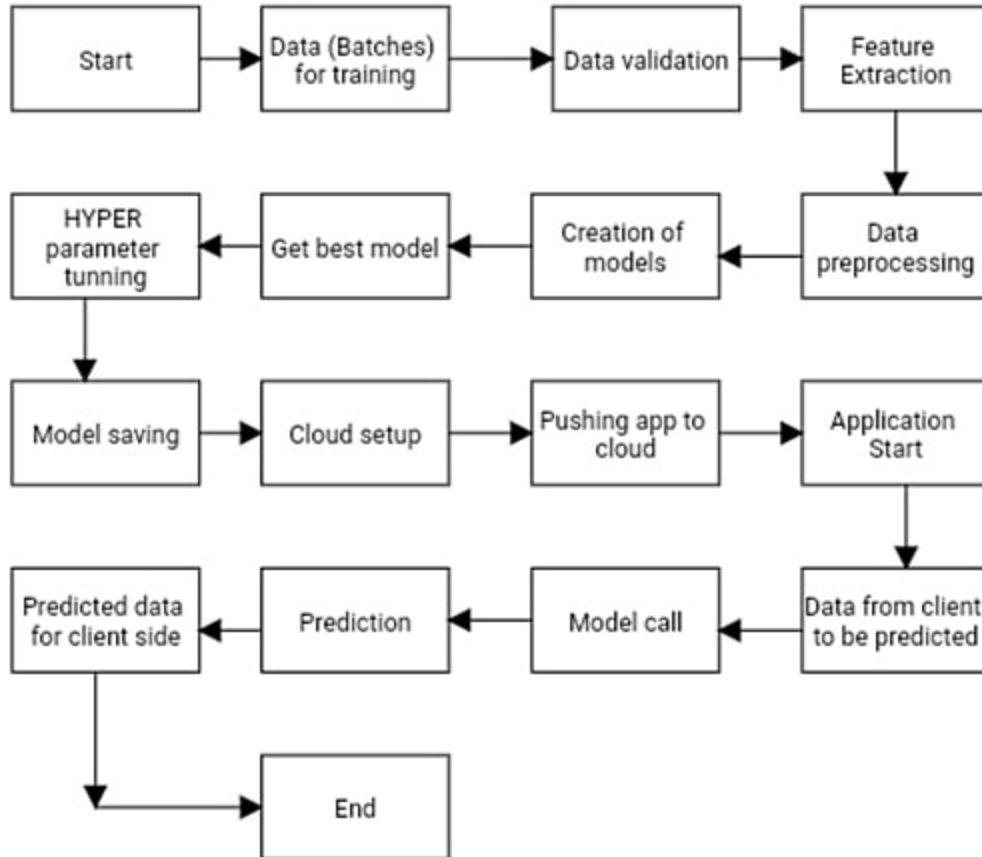
Following are the non-functional requirements of the proposed solution.

FR No.	Non- Fuctional Requirements	Description
NFR-1	Usability	Usability is commonly considered to be the enemy of security. In general,beingsecure means taking extra steps to avoid falling for different attacks. this is especially true of phishing where the best ways to prevent most

		phishing attacks are commonly known, but cybersecurity guidance is rarely followed.
NFR-2	Security	Implementation of updated security algorithms and techniques.
NFR-3	Reliability	The reliability factor evaluates if a suspected site is legitimate or not.
NFR-4	Performance	A phishing website has two key characteristics: it closely resembles a real website and has at least one field for users to enter their credentials. A suspicious attachment is frequently used as a phishing attempt warning sign.
NFR-5	Availability	A common social engineering tactic used to acquire user credentials is phishing, containing account information and payment information. It happens when an attacker deceives a victim into opening an email, instant message, or text message by disguising themselves as a reliable source.
NFR-6	Scalability	Scalable phishing detection and isolation, the primary ideas are to shift protection from end users to network providers and to use the innovative bad neighbourhood concept to detect and isolate both phishing email and phishing web servers.

## 5.PROJECT DESIGN

### 5.1 Data flow diagram:



## 5.2 Technical Architecture:

---

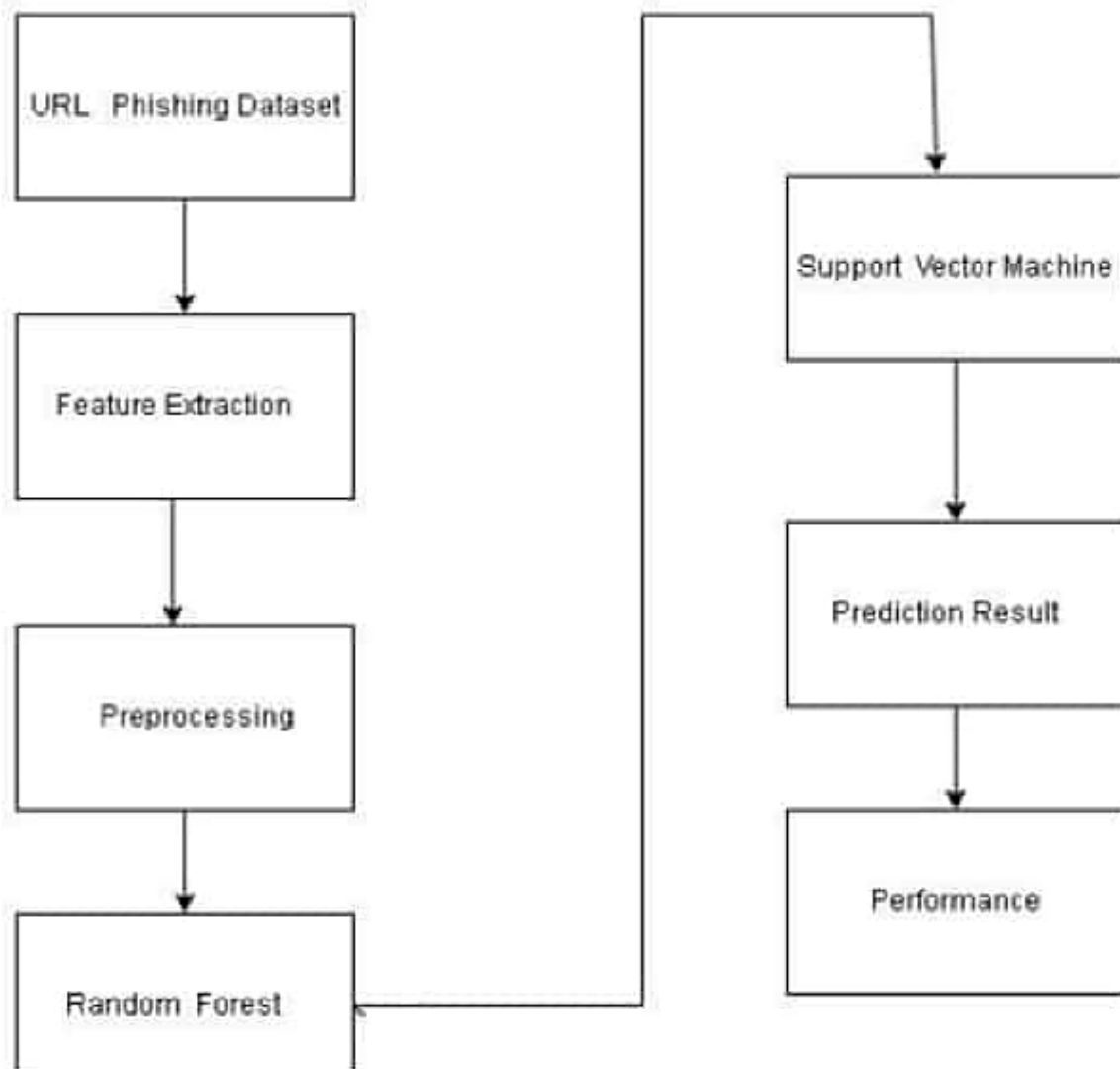


Table-1:Components & Technologies:

S.No	Component	Description	Technology
1.	user interface	how user interacts with application e.g.web UI,mobile.	HTML,CSS,javaScript/ Angular js/React js etc.
2.	application logic-1	logic for a process in the application	java/python
3.	application logic-2	logic for a process in the application	IBM Waston service
4.	dataset	description of data.data preprocessing	csv
5.	cloud	service on cloud	IBM cloud
6.	external API-1	purpose of external API used in the application	python flask API
7.	machine learning model	purpose of machine learning model	random forces:classification
8.	infra structure(server,cloud)	application deployment on local system/cloud local server configuration: cloud server configuration:	local,cloud foundry,kubemetes,e tc

Table-2:Application characteristics:

S. NO	Characteristics	Description	technology
1.	Open-source frameworks	flask is a open source framework used to develop a web -UI and web server	Flask
2.	Security implementations	access to database system is managed by facilities that reside outside the	IBM Service

		database system(authentication)whereas access within the database system is managed by the database manager(authorisation)	
3.	Scalable architecture	application related to python are scaled using flask	Flask,IBM Cloud Object storage
4.	Availability	since this is a web app,the availability for all user ensures that they have internet access.also the cloud storage we have is available easily from ibm	Flask,IBM Cloud Object storage
5.	Performance	the performance is based on the number of users registering for the application.the	Flask,IBM Cloud Object storage

## 6.PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule and Estimation (4 Marks)

Product backlog and sprint schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Homepage	USN-1	As a user, I can explore the resources of the homepage for the functioning	10	Low	Sanjana S AngalaBharathi C Sabitha M
Sprint-1		USN-2	As a user, I can learn about the various sides of the web phishing and be aware of the scams	5	High	Sanjana S , Angala Bharathi C Roshini G
Sprint-2	Final page	USN-3	As a user, I can explore the resources of the final page for the functioning	15	Low	Sabitha M Rohini KS, Roshini G
Sprint-3	Prediction	USN-4	As a user, I can predict the URL easily for detecting whether the website is legitimate or not	10	High	Sanjana S , AngalaBharathi C Sabitha M Rohini KS
	Dashboard					
Sprint-4	Chat	USN-5	As a user, I can share the experience or contact the admin for the support	10	High	Sabitha M Sanjana S, Roshini G, Rohin KS
Sprint-1	Homepage	USN-6	As a admin, we can design interface and maintain the functioning of the website	5	High	Sanjana S Sabitha M Roshini G AngalaBharathi C

Sprint-2	Final page	USN-7	As a admin, we can design the complexity of the website for making it user-friendly	5	Medium	Roshini G Rohini KS
Sprint-3	Prediction	USN-8	As a admin, we can use various ML classifier model for the accurate result for the detection of URL	10	High	Sanjana S , Sabitha B AngalaBharathiC, Rohini KS Roshini G
	Dashboard					
Sprint-4		USN-9	As a admin, we can response to the user message for improvement of the website	10	Medium	Sanjana S Rohini KS, Sabitha M

Project Tracker, Velocity & Burndown Chart (4 Marks)

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 Oct2022	20	29 Oct2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05Nov2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12Nov2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	13Nov2022

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

### 7.1 Feature 1





## 7.2 Feature 2



```
[000] Requirement already satisfied: six==1.15.0 in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from
[001] flask==2.1.0)
[002] Requirement already satisfied: idna==2.5 in /opt/conda/envs/Python3.9/lib/python3.9/site-packages
[003] from requests==2.31.0
[004] Requirement already satisfied: urllib3==2.0.0 in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (f
[005] Requirement already satisfied: sipp==0.5 in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (f
[006] Requirement already satisfied: ipaddress==2.0.0 in /opt/conda/envs/Python3.9/lib/python3.9/site-packag
[007] ibm-watson-machine-learning==3.0.4 in /opt/conda/envs/Python3.9/lib/python3.9/s
[008]
[009]
[010]
[011]
[012]
[013]
[014]
[015]
[016]
[017]
[018]
[019]
[020]
[021]
[022]
[023]
[024]
[025]
[026]
[027]
[028]
[029]
[030]
[031]
[032]
[033]
[034]
[035]
[036]
[037]
[038]
[039]
[040]
[041]
[042]
[043]
[044]
[045]
[046]
[047]
[048]
[049]
[050]
[051]
[052]
[053]
[054]
[055]
[056]
[057]
[058]
[059]
[060]
[061]
[062]
[063]
[064]
[065]
[066]
[067]
[068]
[069]
[070]
[071]
[072]
[073]
[074]
[075]
[076]
[077]
[078]
[079]
[080]
[081]
[082]
[083]
[084]
[085]
[086]
[087]
[088]
[089]
[090]
[091]
[092]
[093]
[094]
[095]
[096]
[097]
[098]
[099]
[100]
[101]
[102]
[103]
[104]
[105]
[106]
[107]
[108]
[109]
[110]
[111]
[112]
[113]
[114]
[115]
[116]
[117]
[118]
[119]
[120]
[121]
[122]
[123]
[124]
[125]
[126]
[127]
[128]
[129]
[130]
[131]
[132]
[133]
[134]
[135]
[136]
[137]
[138]
[139]
[140]
[141]
[142]
[143]
[144]
[145]
[146]
[147]
[148]
[149]
[150]
[151]
[152]
[153]
[154]
[155]
[156]
[157]
[158]
[159]
[160]
[161]
[162]
[163]
[164]
[165]
[166]
[167]
[168]
[169]
[170]
[171]
[172]
[173]
[174]
[175]
[176]
[177]
[178]
[179]
[180]
[181]
[182]
[183]
[184]
[185]
[186]
[187]
[188]
[189]
[190]
[191]
[192]
[193]
[194]
[195]
[196]
[197]
[198]
[199]
[200]
[201]
[202]
[203]
[204]
[205]
[206]
[207]
[208]
[209]
[210]
[211]
[212]
[213]
[214]
[215]
[216]
[217]
[218]
[219]
[220]
[221]
[222]
[223]
[224]
[225]
[226]
[227]
[228]
[229]
[230]
[231]
[232]
[233]
[234]
[235]
[236]
[237]
[238]
[239]
[240]
[241]
[242]
[243]
[244]
[245]
[246]
[247]
[248]
[249]
[250]
[251]
[252]
[253]
[254]
[255]
[256]
[257]
[258]
[259]
[260]
[261]
[262]
[263]
[264]
[265]
[266]
[267]
[268]
[269]
[270]
[271]
[272]
[273]
[274]
[275]
[276]
[277]
[278]
[279]
[280]
[281]
[282]
[283]
[284]
[285]
[286]
[287]
[288]
[289]
[290]
[291]
[292]
[293]
[294]
[295]
[296]
[297]
[298]
[299]
[300]
[301]
[302]
[303]
[304]
[305]
[306]
[307]
[308]
[309]
[310]
[311]
[312]
[313]
[314]
[315]
[316]
[317]
[318]
[319]
[320]
[321]
[322]
[323]
[324]
[325]
[326]
[327]
[328]
[329]
[330]
[331]
[332]
[333]
[334]
[335]
[336]
[337]
[338]
[339]
[340]
[341]
[342]
[343]
[344]
[345]
[346]
[347]
[348]
[349]
[350]
[351]
[352]
[353]
[354]
[355]
[356]
[357]
[358]
[359]
[360]
[361]
[362]
[363]
[364]
[365]
[366]
[367]
[368]
[369]
[370]
[371]
[372]
[373]
[374]
[375]
[376]
[377]
[378]
[379]
[380]
[381]
[382]
[383]
[384]
[385]
[386]
[387]
[388]
[389]
[390]
[391]
[392]
[393]
[394]
[395]
[396]
[397]
[398]
[399]
[400]
[401]
[402]
[403]
[404]
[405]
[406]
[407]
[408]
[409]
[410]
[411]
[412]
[413]
[414]
[415]
[416]
[417]
[418]
[419]
[420]
[421]
[422]
[423]
[424]
[425]
[426]
[427]
[428]
[429]
[430]
[431]
[432]
[433]
[434]
[435]
[436]
[437]
[438]
[439]
[440]
[441]
[442]
[443]
[444]
[445]
[446]
[447]
[448]
[449]
[450]
[451]
[452]
[453]
[454]
[455]
[456]
[457]
[458]
[459]
[460]
[461]
[462]
[463]
[464]
[465]
[466]
[467]
[468]
[469]
[470]
[471]
[472]
[473]
[474]
[475]
[476]
[477]
[478]
[479]
[480]
[481]
[482]
[483]
[484]
[485]
[486]
[487]
[488]
[489]
[490]
[491]
[492]
[493]
[494]
[495]
[496]
[497]
[498]
[499]
[500]
[501]
[502]
[503]
[504]
[505]
[506]
[507]
[508]
[509]
[510]
[511]
[512]
[513]
[514]
[515]
[516]
[517]
[518]
[519]
[520]
[521]
[522]
[523]
[524]
[525]
[526]
[527]
[528]
[529]
[530]
[531]
[532]
[533]
[534]
[535]
[536]
[537]
[538]
[539]
[540]
[541]
[542]
[543]
[544]
[545]
[546]
[547]
[548]
[549]
[550]
[551]
[552]
[553]
[554]
[555]
[556]
[557]
[558]
[559]
[560]
[561]
[562]
[563]
[564]
[565]
[566]
[567]
[568]
[569]
[570]
[571]
[572]
[573]
[574]
[575]
[576]
[577]
[578]
[579]
[580]
[581]
[582]
[583]
[584]
[585]
[586]
[587]
[588]
[589]
[590]
[591]
[592]
[593]
[594]
[595]
[596]
[597]
[598]
[599]
[600]
[601]
[602]
[603]
[604]
[605]
[606]
[607]
[608]
[609]
[610]
[611]
[612]
[613]
[614]
[615]
[616]
[617]
[618]
[619]
[620]
[621]
[622]
[623]
[624]
[625]
[626]
[627]
[628]
[629]
[630]
[631]
[632]
[633]
[634]
[635]
[636]
[637]
[638]
[639]
[640]
[641]
[642]
[643]
[644]
[645]
[646]
[647]
[6
```

## 8.TESTING

Project team shall fill the following information model performance using testing template.

### 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	7	0	0	7
Client Application	43	0	0	43
Security	4	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	5	0	0	5

### Prediction results

Prediction type

Binary classification

Prediction percentage

3

Records

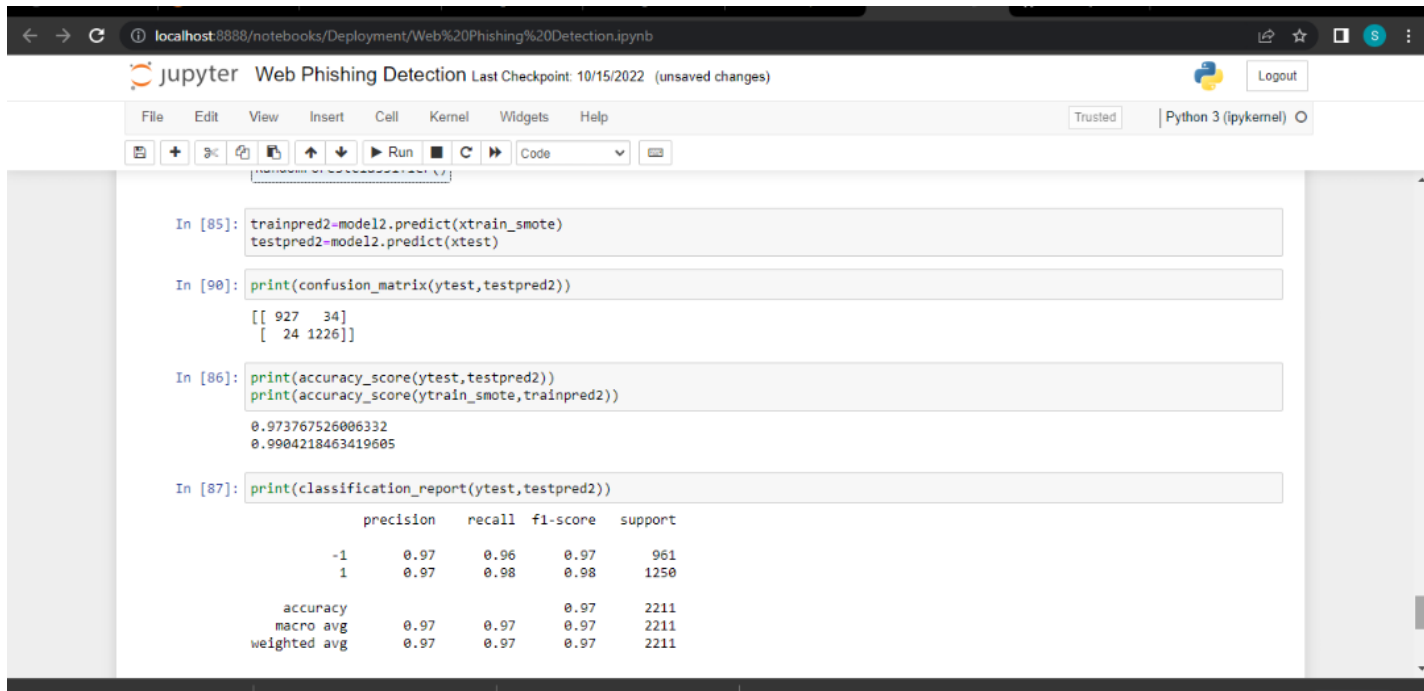
☒ Table view ☐ JSON view

	Prediction	Confidence
1	1	100%
2	1	100%
3	1	100%
4		
5		
6		
7		
8		
9		
10		

Download

## 9.RESULTS

### 9.1 Performance Metrics:



The screenshot shows a Jupyter Notebook interface with the title 'Web Phishing Detection'. The notebook contains several code cells. The first cell (In [85]) shows the prediction of training and testing data. The second cell (In [90]) prints the confusion matrix. The third cell (In [86]) prints the accuracy scores for the training and testing data. The fourth cell (In [87]) prints the classification report.

```
In [85]: trainpred2=model2.predict(xtrain_smote)
testpred2=model2.predict(xtest)

In [90]: print(confusion_matrix(ytest,testpred2))

[[ 927   34]
 [  24 1226]]

In [86]: print(accuracy_score(ytest,testpred2))
print(accuracy_score(ytrain_smote,trainpred2))

0.973767526006332
0.9904218463419605

In [87]: print(classification_report(ytest,testpred2))
```

	precision	recall	f1-score	support
-1	0.97	0.96	0.97	961
1	0.97	0.98	0.98	1250
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

## 10.Advantages

- 1.Measure the degrees of corporate and employee vulnerability.
- 2.Eliminate the cyber threat risk level.
- 3.Increase user alertness to phishing risks.
- 4.Instill a cyber security culture and create cyber security heroes.
- 5.Protect valuable corporate and personal data.
- 6.Meet industry compliance obligation.
- 7.Deploy targeted anti-phishing solutions.
- 8.Segment phishing simulation.

## Disadvantages

- 1.Loss of money
- 2.Loss of intellectual property
- 3.Damage to reputation.
- 4.Disruption of operational activities.
- 5.Blacklist cannot detect new phishing attacks.
- 6.Network based time consuming and costly

7.Content based depends on standard databases.

8.White list positive rate is less.

## 11.Conclusion And Future Scope

Phishing cannot be solved with a single solution.It is acritical situation in which phishers always try to come up with brand new modes of manipulating the consumers.Online consumers should embrace regular risk scrutiny to detect the recent techniques which may head to a thriving phishing attack.To find safer ways,user must be aeare about the dangers of advanced malware which are taking place nowadays.Also,safekeeping teams need to execute advanced methodlogies that can put the advanced threats to an end that are recently beging bypassed by their predictable resentment.Future contribution is done in detecting the identity theft and the phishing mails.It does not involve in the rising trends towards email outsourcing.Log analysis and communication taking place across managerial boundaries can prove to be a tricky one.In other words,we can also say that other electronictransactions will also become a part of the threats.Henceforth,it is suggested to sincerely work on these problems before attacks are beging clutched widly.A command should be acquired which can protect all crucial internet banking activities.

## 12.Appendix

### Source code

#### INDEX.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=0.1">
    <meta name="description" content="This website is develop for identify the safety of URL.">
    <meta name="keywords" content="web phishing detection,phishing url,machine
learning,jupyter notebook,classififer,random forest,cyber security">
    <meta name="authors" content="1.SANJANA S,2.ANGALA BHARATHI C,3.SABITHA
M,4.ROSHINI G,5.ROHINI KS">
    <title> WEB PHISHING DETECTION </title>
    <style>
      body
      {
        width: fit-content; height: fit-content;
        background: url("wpi.jpg");
      }
    </style>
```

```

</head>
<body style="color:lightseagreen ;">
<center>
<div class=" container">
<div class="row">
<div class="form col-md" id="form1">
<h2>WEB PHISHING DETECTION</h2>
<h4> Using Random Forest</h4>

<br>
<form action="/" method ="post">
<label for="url" class="form__label"> <b> URL </b></label>
<input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL"
required="" />
<button class="button" role="button" > <a href="predict.html" target="_blank"> Check
</a></button>
</form>

</div>
</center>
<br>
</div>
</body>

<footer>
<center> <p> &#169 TEAM ID: PNT2022TMID49434 </p> </center>
</footer>
</html>

```

## APP.PY:

```

import numpy as np
import flask
from flask import Flask, request, jsonify, render_template
import pickle
import script

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')

```

```

def predict1():
    return render_template('index.html')

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

@app.route('/y_predict',methods=['POST'])
def y_predict():
    """
    For rendering results on HTML GUI
    """
    url = request.form['URL']
    checkprediction = script.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="Your are safe!! This is a Legitimate Website."

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('index.html', prediction_text='{}'.format(pred),url=url)

@app.route('/predict_api',methods=['POST'])
def predict_api():
    """
    For direct API calls through request
    """
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
if __name__=='__main__':
    app.run(debug=False)

```

## SCRIPT.PY:

```

import datetime
import re

```

```
import socket
import ssl
import urllib.request
```

```
import favicon
import google
import regex
import requests
import whois
import xmltodictPYTHON
from bs4 import BeautifulSoup
from googlesearch import search
from tldextract import extract
```

```
def having_IPhaving_IP_Address(url):
    match=regex.search(
        '(((01)?\d\d\d?|2[0-4]\d|25[0-5])\.\.([01]?\\d\\d?|2[0-4]\d|25[0-5])\.\.([01]?\\d\\d?|2[0-4]\d|25[0-5])\.\.([01]?\\d\\d?|2[0-4]\d|25[0-5])\\V)' #IPv4
        '((0x[0-9a-fA-F]{1,2})\.\.(0x[0-9a-fA-F]{1,2})\.\.(0x[0-9a-fA-F]{1,2})\.\.(0x[0-9a-fA-F]{1,2})\\V)' #IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)    #Ipv6
    if match:
        return -1
    else:
        return 1
```

```
def URLURL_Length (url):
    length=len(url)
    if(length<=75):
        if(length<54):
            return 1
        else:
            return 0
    else:
        return -1
```

```
def Shortining_Service (url):
```

```
match=regex.search('bit\\.ly|goo\\.gl|shorte\\.st|go2l\\.ink|x\\.co|ow\\.ly|t\\.co|tinyurl|tr\\.im|is\\.gd|cli\\.gs|'
```



```

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net',url)
    if match:
        return -1
    else:
        return 1

def having_At_Symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return 1
    else:
        return -1

def double_slash_redirecting(url):
    for i in range(8,len(url)):
        if(url[i]=='/'):

            if(url[i-1]=='/'):
                return -1
    return 1

def Prefix_Suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return -1
    else:
        return 1

def having_Sub_Domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')<=2):
        if(subDomain.count('.')<=1):

```

```
        return 1
    else:
        return 0
else:
    return -1
```

```
def SSLfinal_State(url):
    try:
        response = requests.get(url)
        return 1
    except Exception as e:
        return -1
```

```
def Domain_registration_length(url):
    try:
        domain = whois.whois(url)
        exp=domain.expiration_date[0]
        up=domain.updated_date[0]
        domainlen=(exp-up).days
        if(domainlen<=365):
            return -1
        else:
            return 1
    except:
        return -1
```

```
def Favicon(url):
    subDomain, domain, suffix = extract(url)
    b=domain
    try:
        icons = favicon.get(url)
        icon = icons[0]
        subDomain, domain, suffix =extract(icon.url)
        a=domain
        if(a==b):
            return 1
        else:
            return -1
    except:
        return -1
```

```

def port(url):
    try:
        a_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        location=(url[7:],80)
        result_of_check = a_socket.connect_ex(location)
        if result_of_check == 0:
            return 1
        else:
            return -1
        a_socket.close
    except:
        return -1

```

```

def HTTPS_token(url):
    match=re.search('https://|http://',url)
    if (match.start(0)==0):
        url=url[match.end(0):]
    match=re.search('http|https',url)
    if match:
        return -1
    else:
        return 1

```

```

def Request_URL(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==""):
                linked_to_same = linked_to_same + 1

```

```
vids = soup.findAll('video', src=True)
total = total + len(vids)
```

```
for video in vids:
    subDomain, domain, suffix = extract(video['src'])
    vidDomain = domain
    if(websiteDomain==vidDomain or vidDomain==""):
        linked_to_same = linked_to_same + 1
linked_outside = total-linked_to_same
if(total!=0):
    avg = linked_outside/total
```

```
if(avg<0.22):
    return 1
else:
    return -1
except:
    return -1
```

```
def URL_of_Anchor(url):
```

```
try:
    subDomain, domain, suffix = extract(url)
    websiteDomain = domain

    opener = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(opener, 'lxml')
    anchors = soup.findAll('a', href=True)
    total = len(anchors)
    linked_to_same = 0
    avg = 0
    for anchor in anchors:
        subDomain, domain, suffix = extract(anchor['href'])
        anchorDomain = domain
        if(websiteDomain==anchorDomain or anchorDomain==""):
            linked_to_same = linked_to_same + 1
    linked_outside = total-linked_to_same
    if(total!=0):
        avg = linked_outside/total

    if(avg<0.31):
        return 1
```

```
elif(0.31<=avg<=0.67):  
    return 0  
else:  
    return -1  
except:  
    return 0
```

```
def Links_in_tags(url):  
    try:  
        opener = urllib.request.urlopen(url).read()  
        soup = BeautifulSoup(opener, 'lxml')  
  
        no_of_meta =0  
        no_of_link =0  
        no_of_script =0  
        anchors=0  
        avg =0  
        for meta in soup.find_all('meta'):  
            no_of_meta = no_of_meta+1  
        for link in soup.find_all('link'):  
            no_of_link = no_of_link +1  
        for script in soup.find_all('script'):  
            no_of_script = no_of_script+1  
        for anchor in soup.find_all('a'):  
            anchors = anchors+1  
        total = no_of_meta + no_of_link + no_of_script+anchors  
        tags = no_of_meta + no_of_link + no_of_script  
        if(total!=0):  
            avg = tags/total  
  
        if(avg<0.25):  
            return -1  
        elif(0.25<=avg<=0.81):  
            return 0  
        else:  
            return 1  
    except:  
        return 0
```

```
def SFH(url):
```

```
    return -1
```

```
def Submitting_to_email(url):
```

```
    try:
```

```
        opener = urllib.request.urlopen(url).read()
```

```
        soup = BeautifulSoup(opener, 'xml')
```

```
        if(soup.find('mailto:', 'mail()')):
```

```
            return -1
```

```
        else:
```

```
            return 1
```

```
    except:
```

```
        return -1
```

```
def Abnormal_URL(url):
```

```
    subDomain, domain, suffix = extract(url)
```

```
    try:
```

```
        domain = whois.whois(url)
```

```
        hostname=domain.domain_name[0].lower()
```

```
        match=re.search(hostname,url)
```

```
        if match:
```

```
            return 1
```

```
        else:
```

```
            return -1
```

```
    except:
```

```
        return -1
```

```
def Redirect(url):
```

```
    try:
```

```
        request = requests.get(url)
```

```
        a=request.history
```

```
        if(len(a)<=1):
```

```
            return 1
```

```
        else:
```

```
            return 0
```

```
    except:
```

```
        return 0
```

```
def on_mouseover(url):
```

```
    try:
```

```
opener = urllib.request.urlopen(url).read()
soup = BeautifulSoup(opener, 'lxml')
```

```
no_of_script =0
for meta in soup.find_all(onmouseover=True):
    no_of_script = no_of_script+1
if(no_of_script==0):
    return 1
else:
    return -1
except:
    return -1
```

```
def RightClick(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find_all('script',mousedown=True)):
            return -1
        else:
            return 1
    except:
        return -1
```

```
def popUpWidnow(url):
    return 1
```

```
def Iframe(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        nmeta=0
        for meta in soup.findAll('iframe',src=True):
            nmeta= nmeta+1
        if(nmeta!=0):
            return -1
        else:
            return 1
    except:
        return -1
```

```

def age_of_domain(url):
    try:
        w = whois.whois(url).creation_date[0].year
        if(w<=2018):
            return 1
        else:
            return -1
    except Exception as e:
        return -1

def DNSRecord(url):

    subDomain, domain, suffix = extract(url)
    try:
        dns = 0
        domain_name = whois.whois(url)
    except:
        dns = 1

    if(dns == 1):
        return -1
    else:
        return 1

def web_traffic(url):
    try:
        response = requests.get("http://data.alexa.com/data?cli=10&dat=s&url=" + url)
        dict_data = xmltodict.parse(response.content)
        rank=dict_data['ALEXA']['SD'][1]['REACH']['@RANK']

    except TypeError:
        return -1
    rank= int(rank)
    if (rank<100000):
        return 1
    else:
        return 0

def Page_Rank(url):
    return 1

```



```

def Google_Index(url):
    try:
        subDomain, domain, suffix = extract(url)
        a=domain + '.' + suffix
        query = url
        for j in search(query, tld="co.in", num=5, stop=5, pause=2):
            subDomain, domain, suffix = extract(j)
            b=domain + '.' + suffix
            if(a==b):
                return 1
            else:
                return -1
    except:
        return -1

```

```

def Links_pointing_to_page (url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        count = 0
        for link in soup.find_all('a'):
            count += 1
        if(count>=2):
            return 1
        else:
            return 0
    except:
        return -1

```

```

def Statistical_report (url):
    hostname = url
    h = [(x.start(0), x.end(0)) for x in regex.finditer('https://|http://www.|https://www.|http://www.',
hostname)]
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in regex.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:

```

```

hostname = hostname[:h[0][0]]

url_match=regex.search('at\.\ua|usa\.\cc|baltazarpresentes\.\com\.\br|pe\.\hu|esy\.\es|hol\.\es|swed
dy\.\com|myjino\.\ru|96\.\lt|ow\.\ly',url)
try:
    ip_address = socket.gethostbyname(hostname)

ip_match=regex.search('146\.\112\.\61\.\108|213\.\174\.\157\.\151|121\.\50\.\168\.\88|192\.\185\.\217
\.\116|78\.\46\.\211\.\158|181\.\174\.\165\.\13|46\.\242\.\145\.\103|121\.\50\.\168\.\40|83\.\125\.\22\.\21
9|46\.\242\.\145\.\98|107\.\151\.\148\.\44|107\.\151\.\148\.\107|64\.\70\.\19\.\203|199\.\184\.\144\.\27|1
07\.\151\.\148\.\108|107\.\151\.\148\.\109|119\.\28\.\52\.\61|54\.\83\.\43\.\69|52\.\69\.\166\.\231|216\.\5
8\.\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\.\15
7|34\.\196\.\13\.\28|103\.\224\.\212\.\222|172\.\217\.\4\.\225|54\.\72\.\9\.\51|192\.\64\.\147\.\141|198\.\2
00\.\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\.\8
2\.\156\.\19|37\.\157\.\192\.\102|204\.\11\.\56\.\48|110\.\34\.\231\.\42',ip_address)
except:
    return -1

if url_match:
    return -1
else:
    return 1

def main(url):

    check = [[having_IPhaving_IP_Address
(url),URLURL_Length(url),Shortining_Service(url),having_At_Symbol(url),

double_slash_redirecting(url),Prefix_Suffix(url),having_Sub_Domain(url),SSLfinal_State(url),

Domain_registration_length(url),Favicon(url),port(url),HTTPS_token(url),Request_URL(url),

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abnormal_URL(url),
    Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(url),
    age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Google_Index(url),

```

```
Links_pointing_to_page(url),Statistical_report(url)]]
```

```
print(check)
```

```
return check
```

**Git Hub Link**

**Demo Link**