WEB PHISHING DETECTION

TEAM ID:PNT2022TMID43653

AN IBM PROJECT REPORT

SUBMITTED BY

GREESHMA M NAIR - 721419104016

ANUPAMA K - 721419104006

MRIDANI M B - 721419104025

SREERAG K DAS - 721419104046

CHAPTER 1

INTRODUCTION

Introduction to Web Phishing

Phishing is a type of social engineering where an attacker sends a fraudulent (e.g., spoofed, fake, or otherwise deceptive) message designed to trick a person into revealing sensitive information to the attacker or to deploy malicious software on the victim's infrastructure like ransomware. Phishing attacks have become increasingly sophisticated and often transparently mirror the site being targeted, allowing the attacker to observe everything while the victim is navigating the site, and transverse any additional security boundaries with the victim. As of 2020, phishing is by far the most common attack performed by cybercriminals, the FBI's Internet Crime Complaint Centre recording over twice as many incidents of phishing than any other type of computer crime.

The first recorded use of the term "phishing" was in the cracking toolkit AOHell created by Koceilah Rekouche in 1995; however, it is possible that the term was used before this in a print edition of the hacker magazine *2600*.The word is a variant of *fishing*, influenced by phreaking, and alludes to the use of increasingly sophisticated lures to "fish" for users' sensitive information.

Attempts to prevent or mitigate the impact of phishing incidents include legislation, user training, public awareness, and technical security measures. Phishing awareness has become important at home and at the work place. For instance, from 2017 to 2020, phishing attacks have increased from 72% to 86% among businesses.

1.1 PROJECT OVERVIEW

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.

Common threats of web phishing:

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

In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

1.2 PURPOSE

The purpose of this project is to predict phishing website using the data from Kaggle Dataset, and compare different classification models. There have been famous detection problems such as Credit card Fraud Detection, while people have not done great phishing detection because they don't have data with enough attributes. The provided dataset includes 11430 URLs with 87 extracted features. The dataset is designed to be used as benchmarks for machine learning-based phishing detection systems. Features are from three different classes: 56 extracted from the structure and syntax of URLs, 24 extracted from the content of their correspondent pages, and 7 are extracted by querying external services. The dataset is balanced, it contains exactly 50% phishing and 50% legitimate URLs.The gradient boosting algorithm Often provides predictive accuracy that cannot be trumped and can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible. We finally proved that these features are important and good for classification accuracy.Gradient Boosting has repeatedly proven to be one of the most powerful technique to build predictive models in both classification and regression. Because of the fact that Grading Boosting algorithms can easily overfit on a training data set, different constraints or regularization methods can be utilized to enhance the algorithm's performance and combat overfitting.

CHAPTER 2

LITERATURE SURVEY

In emerging technology, industry, which deeply influence today’s security problems, has given a headache to many employers and home users. Occurrences that exploit human vulnerabilities have been on the upsurge in recent years. In these new times there are many security systems being enabled to ensure security is given the outmost priority and prevention to be taken from being hacked by those who are involved in cyber-offenses and essential prevention is taken as high importance in organization to ensure network security is not being compromised. Cyber security employee are currently searching for trustworthy and steady detection techniques for phishing websites detection. Due to wide usage of internet to perform various activities such as online bill payment, banking transaction, online shopping, etc. Customer face numerous security threats like cybercrime. Many cybercrime is being casually executed for example spam, fraud, identity theft cyber terrorisms and phishing. Among this phishing is known as the most common cybercrime today. Phishing has become one amongst the top three most current methods of law breaking in line with recent reports, and both frequency of events and user weakness has increased in recent years, more combination of all these methods result in greater danger of economic damage. Phishing is a social engineering attack that targets and exploiting the weakness found in the system at the user’s end. This paper proposes the Agile Unified Process (AUP) to detect duplicate websites that can potentially collect sensitive information about the user. The system checks the blacklisted sites in dataset and learns the patterns followed by the phishing websites and applies it to further given inputs. The system sends a pop-up and an e-mail notification to the user, if the user clicks on a phishing link and redirects to the site if it is a safe website. This system does not support real time detection of phishing sites; user has to supply the website link to the system developed with Microsoft Visual Studio 2010 Ultimate and MySQL stocks up data and to implement database in this system. Phishing costs Internet user’s lots of money. It refers to misusing weakness on the user side, which is vulnerable to such attacks. The basic ideology of the proposed solution is use to all the three-hybrid solution blacklist and whitelist, heuristics and visual similarity. The proposed system carries out a set of procedures before giving out the results. First, it tracks all “http” traffic of client system by creating a browser extension. Then compare domain of each URL with the white list of trusted domains and the blacklist of illegitimate domains.

Further various characters in the URL is considered like number of ‘@’, number of ‘-

‘and many more. Next approach is to extract and compare CSS of doubtful URL and compare it with the CSS of each of the legitimate domains in queue. This method will look into visual based features of the phished websites and machinelearning classifiers such as decision tree, logistic regression, random forest are applied to the collected data, and a score is generated. The match score and similarity score is evaluated. If the score is greater than threshold then the URL marked as phishing and blocked. This approach provides a three level security block. Phishing is a dangerous effort to steal private data from users like address, Aadhar number, PAN card details, credit or debit card details, bank account details, personal details etc. The various types of phishing attacks like spoofing, instant spam spoofing, Hosts file poisoning, malware-based phishing, Manin-the middle, session hijacking, DNS based phishing, deceptive phishing, key loggers/loggers, Web Trojans, Data theft, Content-injection phishing, Search engine phishing, Email /Spam, Web based delivery, Link Manipulation, System reconfiguration, Phone phishing, etc.

2.1 EXISTING PROBLEMS

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users. Unfortunately, many of the existing phishingdetection tools, especially those that depend on an existing blacklist, suffer limitations such as low detection accuracy and high false alarm that is often caused by either a delay in blacklist update as a result of human verification process involved in classification or perhaps, it can be attributed to human error in classification which may lead to improper classification of the classes.

These critical issues have drawn many researchers to work on various approaches to improve detection accuracy of phishing attacks and to minimize false alarm rate. The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model. Therefore, it requires an effective technique to regulate retraining as to enable machine learning algorithm to actively adapt to the changes in phish patterns. The ML based phishing techniques depend on website functionalities to gather information that can help classify websites for detecting phishing sites. The problem of phishing cannot be eradicated, nonetheless can be reduced by combating it in two ways, improving targeted antiphishing procedures and techniques and informing the public on how fraudulent phishing websites can be detected and identified.

2.2 REFERENCES

Detecting Phishing Websites Using Machine Learning by Sagar Patil, Yogesh Shetye, Nilesh Shendage published in the year 2020.

Machine Learning-Based Phishing Attack Detection by Sohrab Hossain, Dhiman Sarma, Rana Joythi Chakma published in the year 2020.

Phishing website detection based on effective machine learning approach by Gururaj Harinahalli Lokesh published in the year 2020.

Research on Website Phishing Detection Based on LSTM RNN by Yang Su published in the year 2020.

Detecting Phishing Website Using Machine Learning by Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Asif Iqbal Hajamydeen published in the year 2020.

Fette, I., Sadeh, N.M., Tomasic, A. "Learning to detect phishing emails." In Proceedings of the 16th International Conference on World Wide Web (WWW'07), May 2017.

Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. "A comparison of machine learning techniques for phishing detection." In Proceedings of eCrime Researchers Summit (eCryme '07), Oct 2010.

Basnet, R., Mukkamala, S., Sung, A.H. "Detection of phishing attacks: A machine learning approach." Studies in Fuzziness and Soft Computing, 226:373–383, 2014.

Lakshmi, V. Santhana, and M. S. Vijaya. "Efficient prediction of phishing websites using supervised learning algorithms." *Procedia Engineering* 30 (2012): 798-805.

Ramzan, Zulfikar (2010). "Phishing attacks and countermeasures". In Stamp, Mark; Stavroulakis, Peter (eds.). Handbook of Information and Communication Security. Springer.

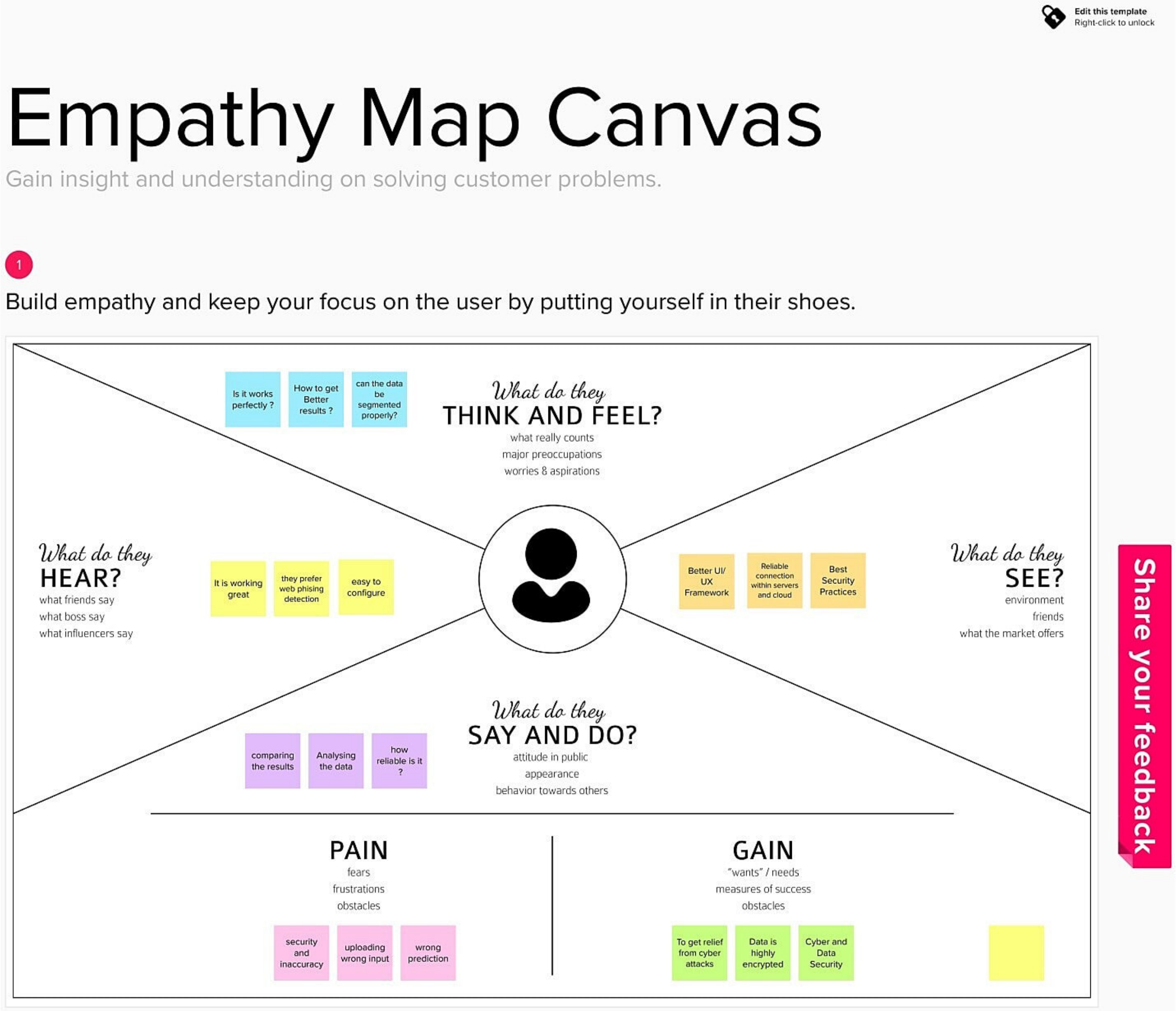
2.3 PROBLEM STATEMENT DEFINITION

A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state and any gaps between the two. A problem statement is an important communication tool that can help ensure everyone working on a project knows what the problem they need to address is and why the project is important.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



Edit this template
Right-click to unlock

Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1

Build empathy and keep your focus on the user by putting yourself in their shoes.

What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations

Is it works perfectly ?

How to get Better results ?

can the data be segmented properly?

What do they HEAR?
what friends say
what boss say
what influencers say

It is working great

they prefer web phising detection

easy to configure

What do they SEE?
environment
friends
what the market offers

Better UI/ UX Framework

Reliable connection within servers and cloud

Best Security Practices

What do they SAY AND DO?
attitude in public
appearance
behavior towards others

comparing the results

Analysing the data

how reliable is it ?

PAIN
fears
frustrations
obstacles

security and inaccuracy

uploading wrong input

wrong prediction

GAIN
"wants" / needs
measures of success
obstacles

To get relief from cyber attacks

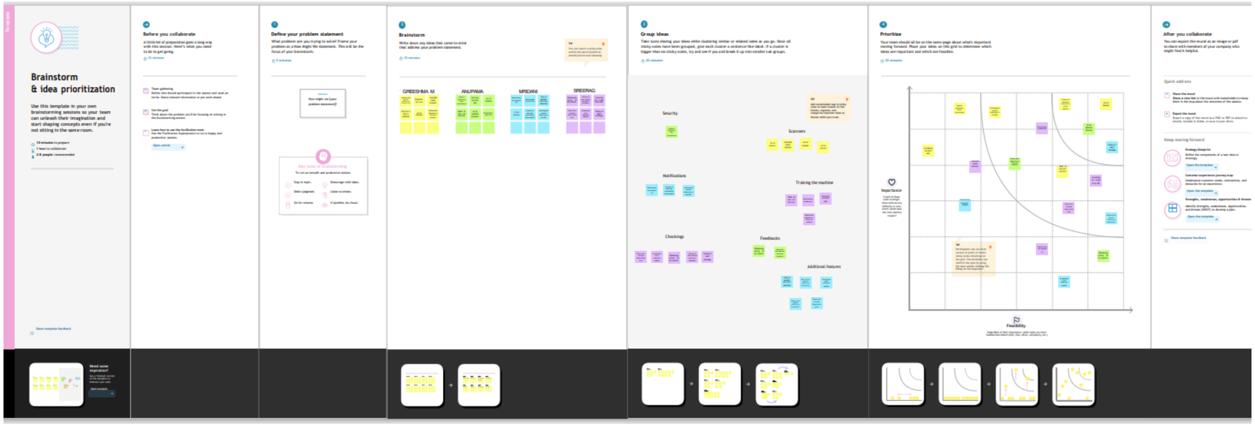
Data is highly encrypted

Cyber and Data Security

Share your feedback

Figure 3.1 Empathy Map Canvas

3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

Table 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To reduce the people falling for web phishing scams by creating a sophisticated tool that classifies a website as malicious or safe to use.
2.	Idea / Solution description	Our solution is to build an efficient and intelligent system to detect phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.
3.	Novelty / Uniqueness	Uses an Ensemble model ,Explores weighted features for Neural Network approaches, Extensive feature extraction strategy from the URL Simple and Easy-to-Understand UI.
4.	Social Impact / Customer Satisfaction	By using this application the customer has the sense of safety whenever he attempts to provide sensitive information to a site.
5.	Business Model (Revenue Model)	This developed model can be used as an enterprise applications by organizations which handles sensitive information and also can be sold to government agencies to prevent the loss of potential important data.

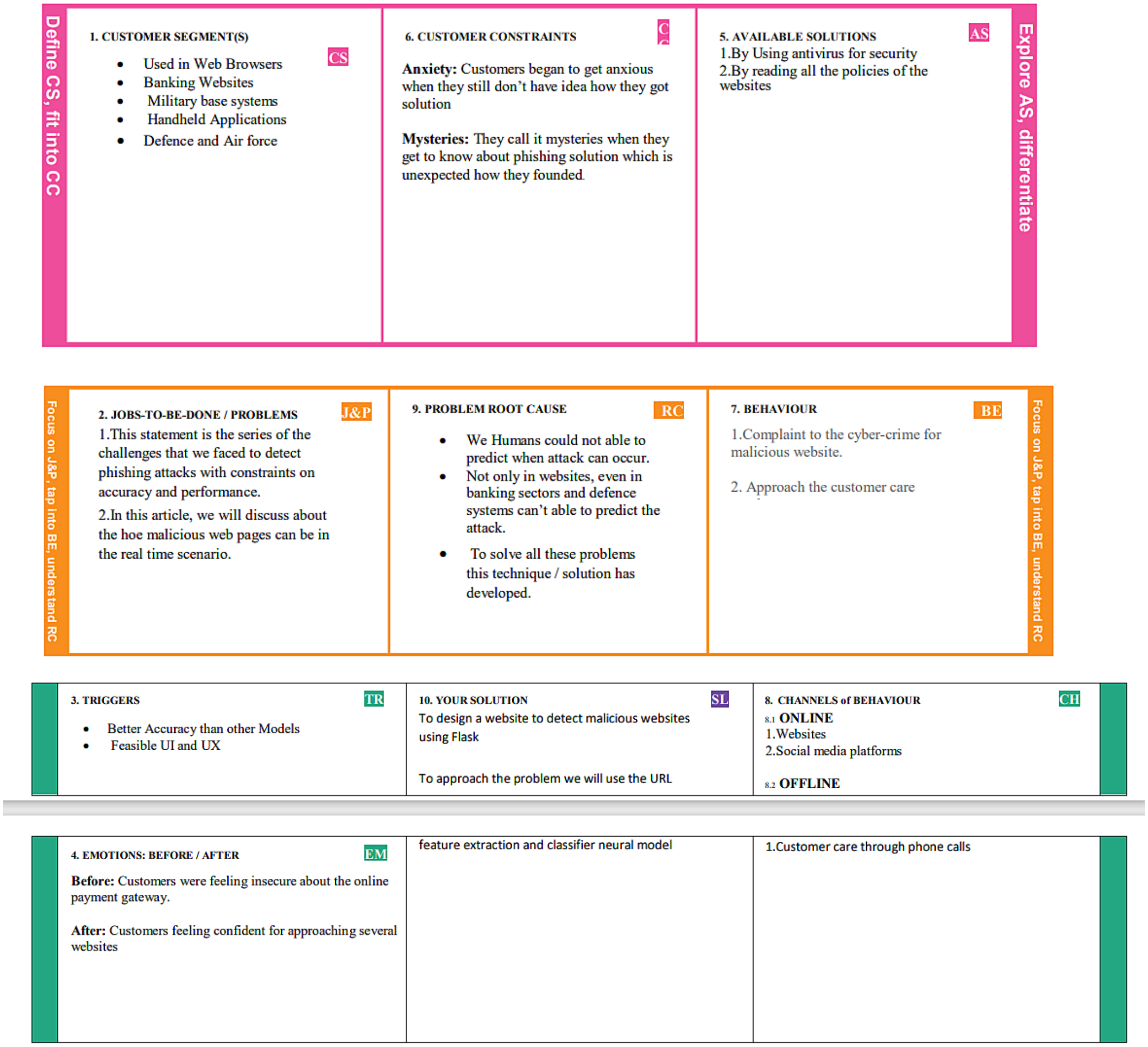
6.	Scalability of the Solution	Solution can use additional hardware resources when the amount of users and activity is increased .The API can ensure that multiple requests at the same time are handled in a parallel fashion.

3.4 PROBLEM SOLUTION FIT

Problem/solution fit is the very beginning stage of a startup when founders are using design thinking to shape their business idea and validate it with their users.

For tech and tech-enabled startups there are several steps that are usually completed at this stage:

1. Problem research - it includes customer segments, pain points for each customer segment and value proposition;
2. Articulating the solution - that includes the product channels, the marketing channels, and the revenue streams;
3. Validating solution hypothesis with potential users - user research
4. PoC development - it can vary from only high-level wireframes to some outcome of the “no code required” platforms.
5. User testing
6. Clickable prototype using available prototyping tools like **InVision** or a more advanced version of the “no code required” platforms mentioned above.



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

6. CUSTOMER CONSTRAINTS CC

Anxiety: Customers began to get anxious when they still don't have idea how they got solution

Mysteries: They call it mysteries when they get to know about phishing solution which is unexpected how they founded.

5. AVAILABLE SOLUTIONS AS

1.By Using antivirus for security
2.By reading all the policies of the websites

Explore AS, differentiate

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

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

1.This statement is the series of the challenges that we faced to detect phishing attacks with constraints on accuracy and performance.

2.In this article, we will discuss about the hoe malicious web pages can be in the real time scenario.

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 defence systems can't able to predict the attack.
- To solve all these problems this technique / solution has developed.

7. BEHAVIOUR BE

1.Complaint to the cyber-crime for malicious website.

2. Approach the customer care

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

3. TRIGGERS TR

- Better Accuracy than other Models
- Feasible UI and UX

10. YOUR SOLUTION SL

To design a website to detect malicious websites using Flask

To approach the problem we will use the URL feature extraction and classifier neural model

8. CHANNELS of BEHAVIOUR CH

8.1 **ONLINE**
1.Websites
2.Social media platforms

8.2 **OFFLINE**
1.Customer care through phone calls

4. EMOTIONS: BEFORE / AFTER EM

Before: Customers were feeling insecure about the online payment gateway.

After: Customers feeling confident for approaching several websites

Figure 3.4 Problem Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

Solution Requirements (Functional & Non-functional)

TABLE 4.1 FUNCTIONAL REQUIREMENT

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
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Password reset	Password reset links and information about its confirmed email address.
FR-4	User cancellations	Cancellation via form
FR-5	Discovery of phishing website	the user with a visual alert sound and vibration alert Notification through email.
FR-6	User Feedback	Feedback through Forms

TABLE 4.2 NON-FUNCTIONAL REQUIREMENTS

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Responsive UI / UX Design and users can easily configure the settings based on their preference.
NFR-2	Security	Implementation of Updated security algorithms and techniques.
NFR-3	Reliability	Reliability Factor determines the possibility of a suspected site to be Valid or Fake.
NFR-4	Performance	The two main characteristics of a phishing site are that it looks extremely similar to a legitimate site and that it has at least one field to enable users to input their credentials.

NFR-5	Availability	It occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.
NFR-6	Scalability	Scalable detection and isolation of phishing, the main ideas are to move the protection from end users towardsthe network provider and to employ the novel bad neighbourhood concept, in order to detect and isolate both phishing e mail senders and phishing web servers.

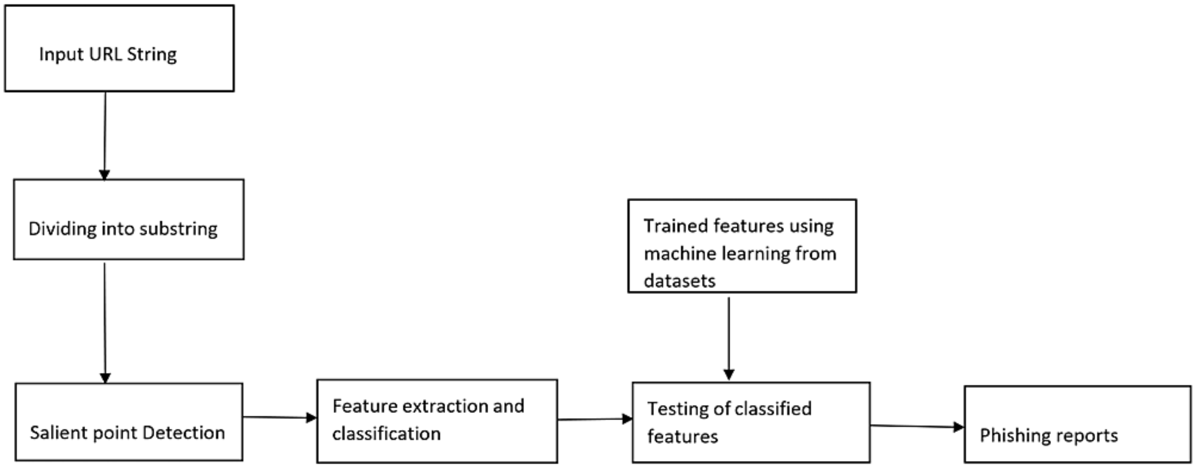
CHAPTER 5

PROJECT DESIGN

Project Design Phase-I

5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically.


Input URL String
Dividing into substring
Trained features using machine learning from datasets
Salient point Detection
Feature extraction and classification
Testing of classified features
Phishing reports

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

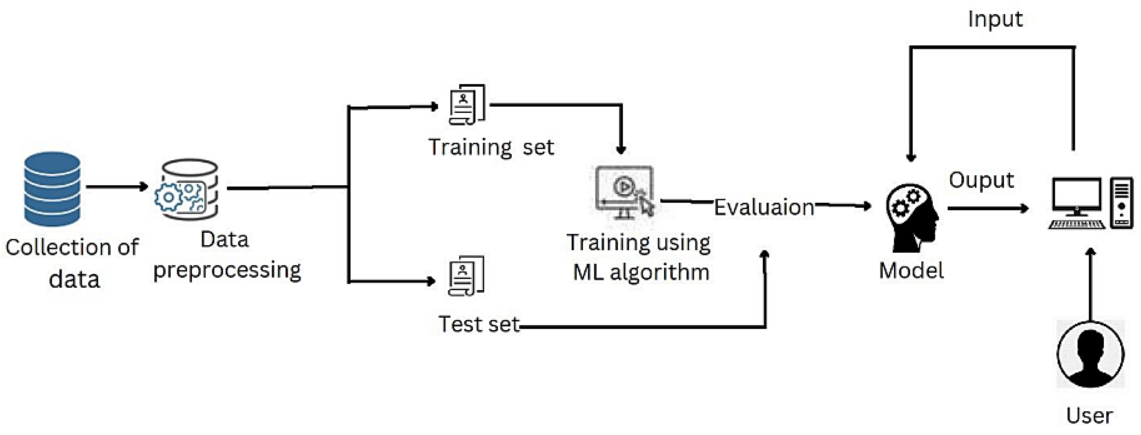
User Type	Functional Requirement	User Story	User Story / Task	Acceptance criteria	Priority	Release

	(Epic)	**Number**				
Customer (C- suite executive, CEO, mobile user, web user)	Detect and predict phishing websites	USN-1	As a user, I to detect phishing websites	I can protect my personal data getting stolen	High	Sprint-1
Customer (C- suite executive, CEO, mobile user, web user)	Identify Fraudulent URL	USN-2	As a user, I need to identify the URL that looks suspicious	I can protect my data from hackers	High	Sprint-2
Customer (C- suite executive, CEO, mobile user, web user)	Identification of valid or invalid URL	USN-3	As a user, I need to identify whether a URL is valid or not	I can prevent online money theft	High	Sprint-2
Customer (C- suite executive, CEO, mobile user, web user)	Identification of accuracy level of detected phished domains	USN-4	As a user, I need to know the accuracy level of detected phished domains	I can ensure safety	Medium	Sprint-3
Customer (C- suite executive, CEO, mobile user, web user)	Identify false positives and false negatives	USN-5	As a user, I need to identify false positives and false negatives	I can prevent unwanted malware	Low	Sprint-4

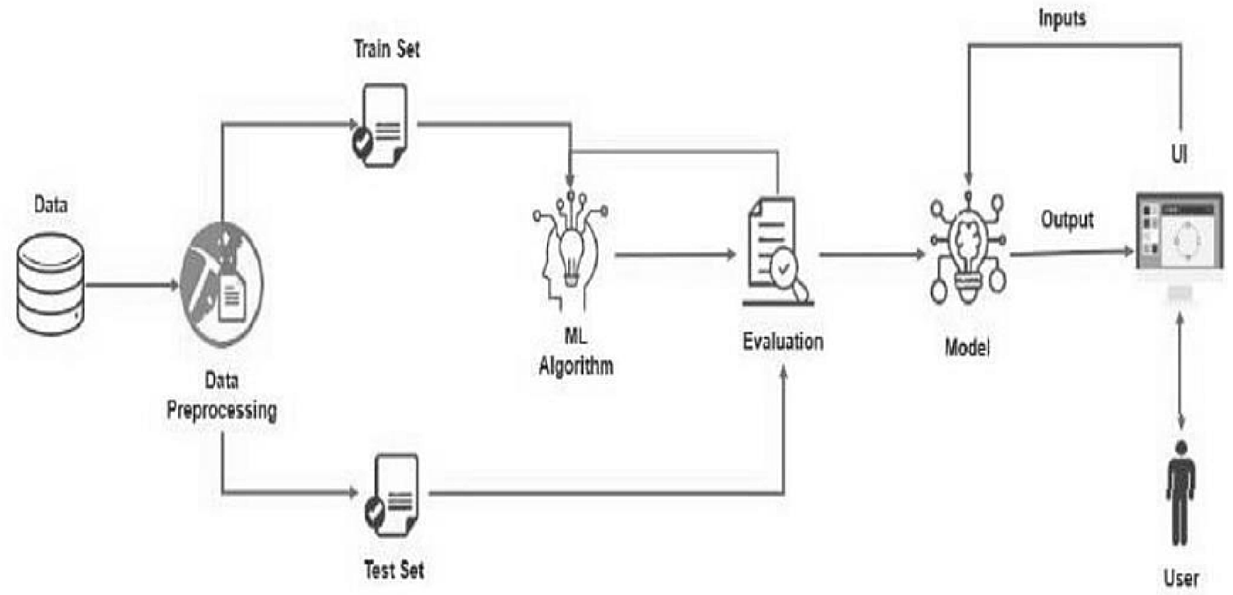
Project Design Phase-II

SOLUTION AND TECHNICAL ARCHITECTURE

SOLUTION ARCHITECTURE:



TECHNOLOGY ARCHITECTURE:



5.3 USER STORIES

User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.

User stories are written by or for users or customers to influence the functionality of the system being developed. In some teams, the product manager (or product owner in Scrum), is primarily responsible for formulating user stories and organizing them into a product backlog. In other teams, anyone can write a user story. User stories can be developed through discussion with stakeholders, based on personas or are simply made up.

A user story is the smallest unit of work in an agile framework. It’s an end goal, not a feature, expressed from the software user’s perspective. In software development and product management, a user story is an **informal, natural language description of features of a software system**.

They are written from the perspective of an end user or user of a system, and may be recorded on index cards, Post-it notes, or digitally in project management software.

CHAPTER 6

PROJECT PLANNING & SCHEDULING

Project Planning (Product Backlog, Sprint Planning, Stories, Story points)

SPRINT PLANNING & ESTIMATION

A sprint schedule is a written description of the entire sprint planning process. It's one of the initial steps in the agile sprint planning process, and it calls for sufficient investigation, preparation, and coordination. It centres on a product backlog, which is a list of open requests for development and iteration

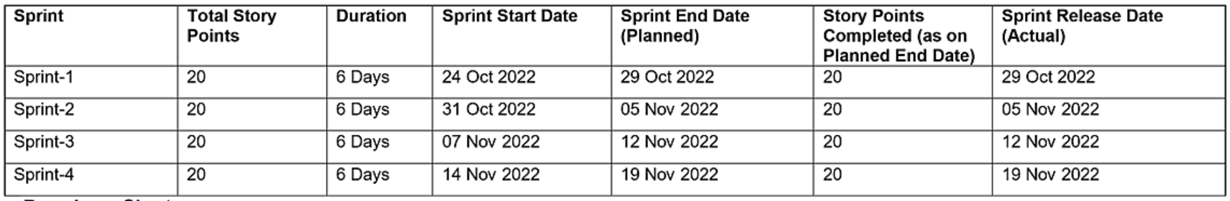
TABLE 6.1

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members

Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	ANUPAMA
Sprint-1	conformation Mail	USN-2	As a user, I can register for the application through Gmail	2	Medium	GREESHMA M NAIR
Sprint-2	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	MRIDANI
Sprint-3	Dashboard	USN-4	As a user, I can logout or change password	1	Medium	SREERAG K DAS
Sprint-4	User input	USN-5	As a user I can input the particular URL in the required field and wait for validation.	2	High	ANUPAMA
Sprint-4	Prediction	USN-6	I will predict the URL websites using Machine Learning algorithms	2	High	MRIDANI

TABLE 6.2 SPRINT DELIVERY SCHEDULE

PROJECT TRACKER AND VELOCITY



Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

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)

We have a 6-day sprint duration, and the velocity of the team is

20 (points per sprint). So our team’s average velocity (AV) per iteration unit (story points per day)

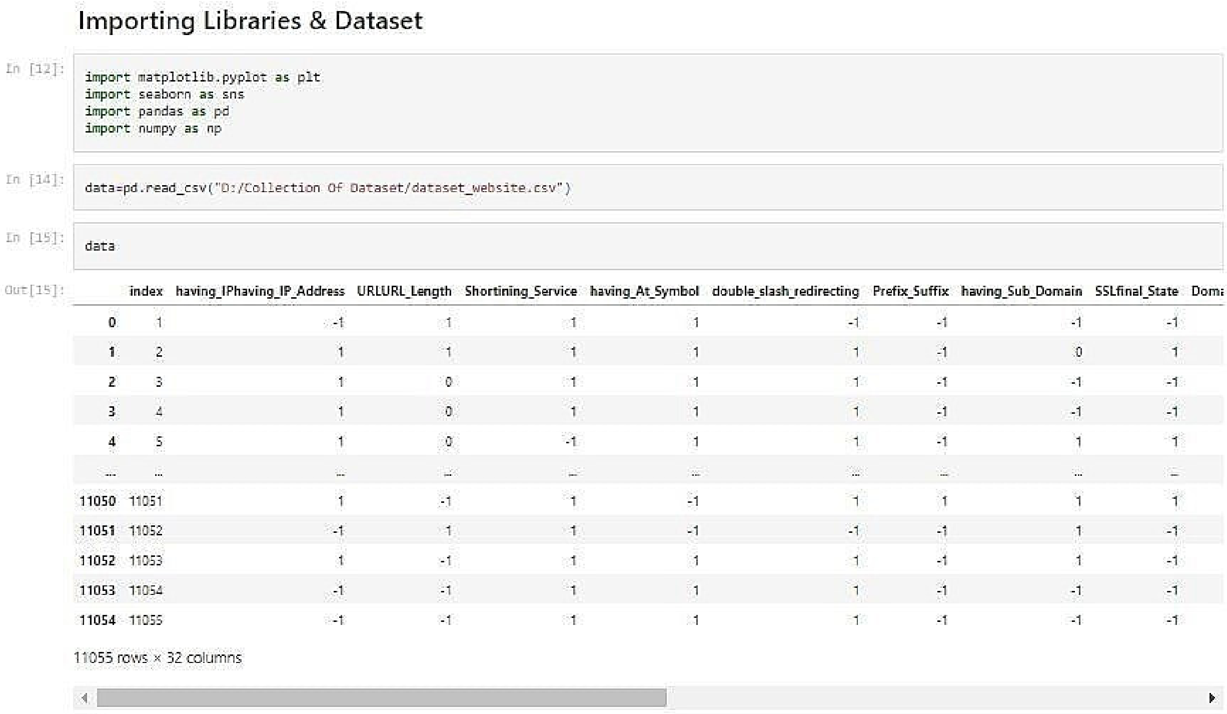
AV = (Sprint Duration / Velocity) = 20 /6 = 3.33

CHAPTER 7

CODING & SOLUTIONING

7.1 FEAUTRE 1

DATA PREPROCESSING & DATA VISUALIZATION



Importing Libraries & Dataset

In [12]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

In [14]:

```
data=pd.read_csv("D:/Collection Of Dataset/dataset_website.csv")
```

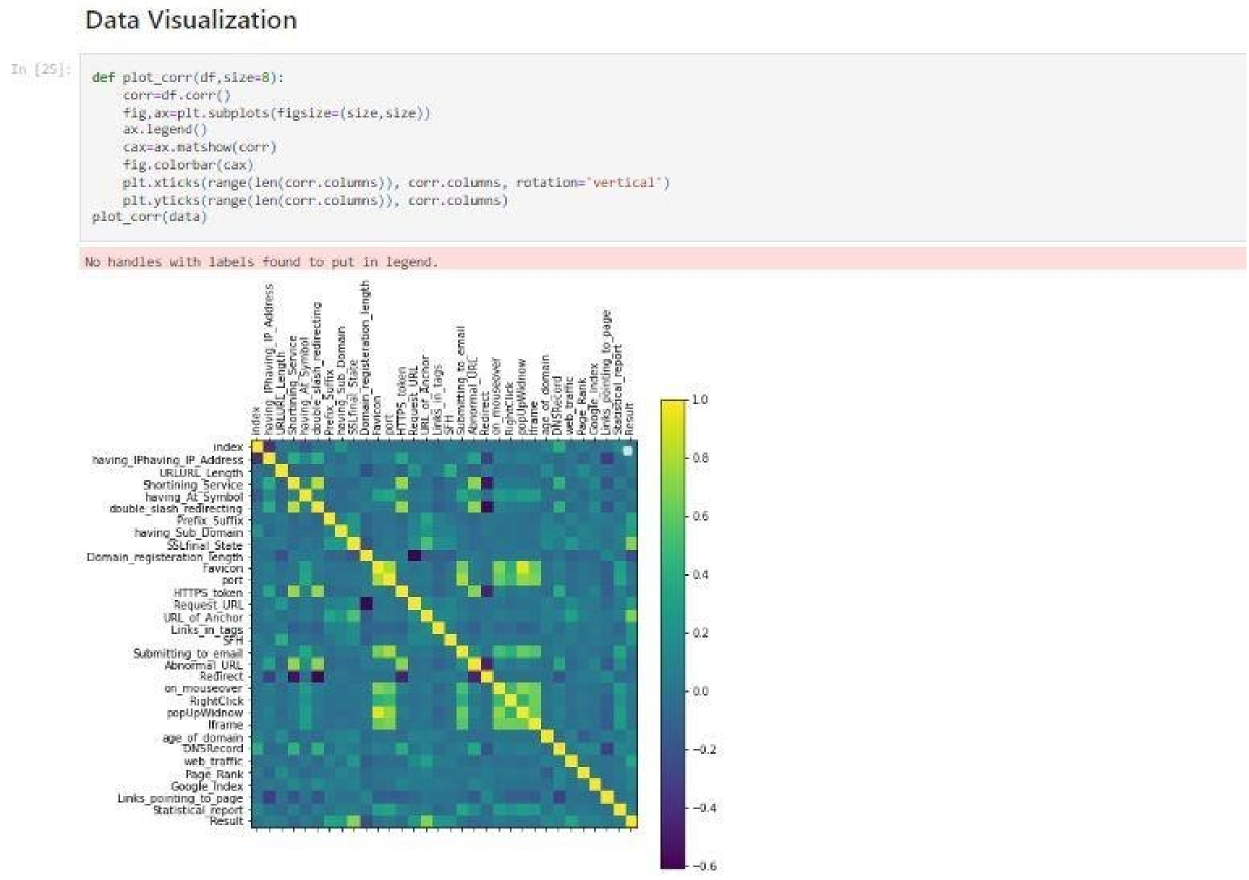
In [15]:

```
data
```

Out[15]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Doma
0	1	-1	1	1	1	-1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	1	
2	3	1	0	1	1	1	-1	-1	-1	
3	4	1	0	1	1	1	-1	-1	-1	
4	5	1	0	-1	1	1	-1	1	1	
...	...	...	...	...	...	...	...	...	...	
11050	11051	1	-1	1	-1	1	1	1	1	
11051	11052	-1	1	1	-1	-1	-1	1	-1	
11052	11053	1	-1	1	1	1	-1	1	-1	
11053	11054	-1	-1	1	1	1	-1	-1	-1	
11054	11055	-1	-1	1	1	1	-1	-1	-1	

11055 rows × 32 columns



Data Visualization

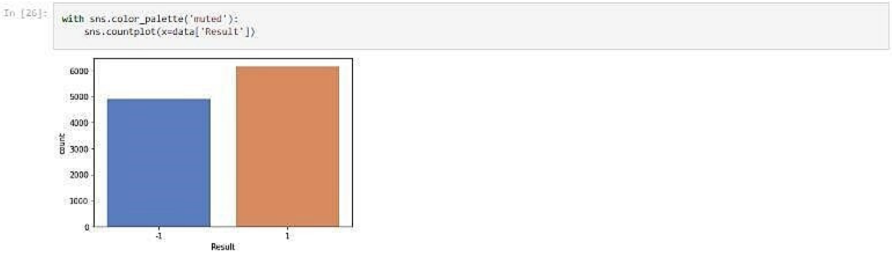
In [25]:

```
def plot_corr(df,size=8):
    corr=df.corr()
    fig,ax=plt.subplots(figsize=(size,size))
    ax.legend()
    cax=ax.matshow(corr)
    fig.colorbar(cax)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical')
    plt.yticks(range(len(corr.columns)), corr.columns)
plot_corr(data)
```

```
No handles with labels found to put in legend.
```

1. There are 11054 instances and 31 fearures in dataset.
2. Out of which 30 are independent features where as 1 is dependent feature.
3. Each feature is in int datatype, so there is no need to use LabelEncoder.
4. There is no outlier present in dataset.
5. There is no missing value in dataset

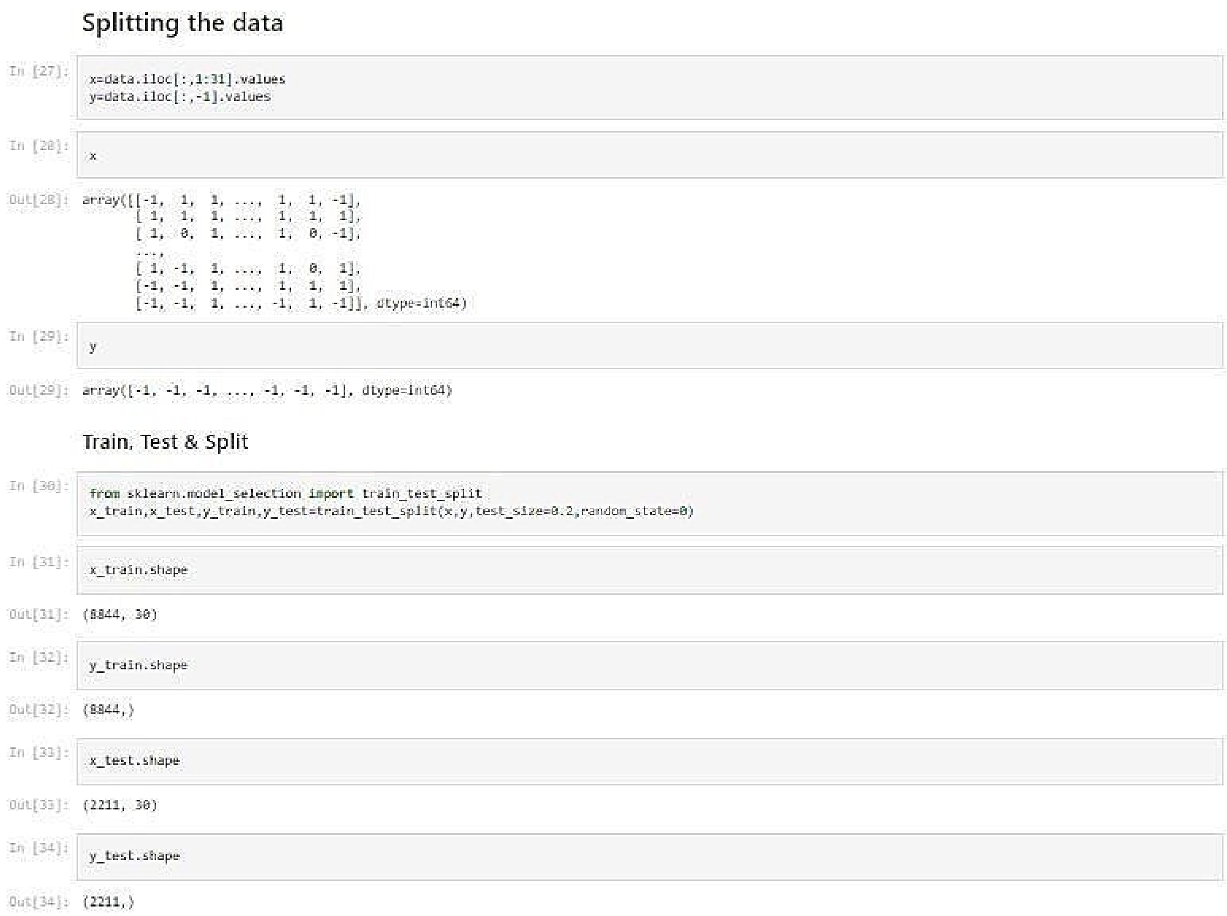
From below image we can infer that in the dataset contains 5000+ Phishing Websites and 6000+ Legitimate Website Feautres.



In [26]:

```
with sns.color_palette('muted'):
    sns.countplot(x=data['Result'])
```

SPLITTING THE DATA



Splitting the data

```
In [27]: x=data.iloc[:,1:31].values
         y=data.iloc[:,-1].values

In [28]: x

Out[28]: array([[-1,  1,  1, ...,  1,  1, -1],
                [ 1,  1,  1, ...,  1,  1,  1],
                [ 1,  0,  1, ...,  1,  0, -1],
                ...,
                [ 1, -1,  1, ...,  1,  0,  1],
                [-1, -1,  1, ...,  1,  1,  1],
                [-1, -1,  1, ..., -1,  1, -1]], dtype=int64)

In [29]: y

Out[29]: array([-1, -1, -1, ..., -1, -1, -1], dtype=int64)
```

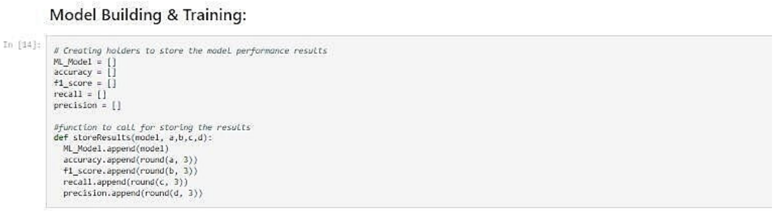
Train, Test & Split

```
In [30]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

In [31]: x_train.shape

Out[31]: (8844, 30)

In [32]: y_train.shape

Out[32]: (8844,)

In [33]: x_test.shape

Out[33]: (2211, 30)

In [34]: y_test.shape

Out[34]: (2211,)
```

CHOOSING THE APPROPRIATE MODEL

Model selection is a key step in every data science project and requires perhaps the most conceptual foundational knowledge.

We'd reviewed a number of supervised machine learning models in class like Logistic Regression, K-Nearest Neighbors, Naive Bayes, Random Forest, and Gradient Boost. Here we used to choose Gradient Boosting Algorithm for predicting best accuracy than other models.



Model Building & Training:

```
In [14]: # Creating holders to store the model performance results
         ML_Model = []
         accuracy = []
         f1_score = []
         recall = []
         precision = []

         #function to call for storing the results
         def storeResults(model, a,b,c,d):
           ML_Model.append(model)
           accuracy.append(round(a, 3))
           f1_score.append(round(b, 3))
           recall.append(round(c, 3))
           precision.append(round(d, 3))
```

CLASSIFICATION REPORT OF THE MODEL:

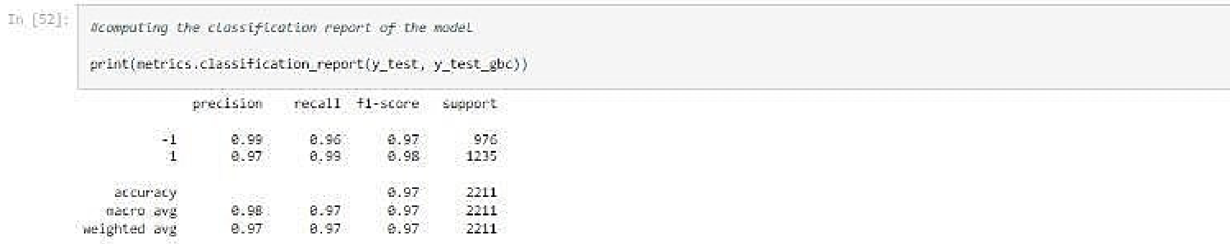
It is one of the performance evaluation metrics of a classification-based machine learning model. It displays your model's precision, recall, F1 score and support. It provides a better understanding of the overall performance of our trained model.

Precision - Precision is defined as the ratio of true positives to the sum of true and false positives.

Recall - Recall is defined as the ratio of true positives to the sum of true positives and false negatives.

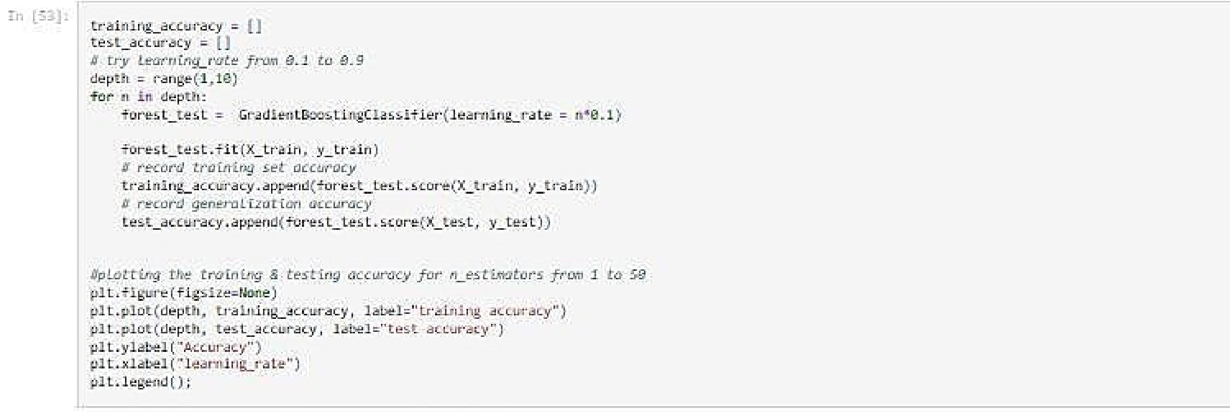
F1 Score - The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.

Support - Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models, it just diagnoses the performance evaluation process.

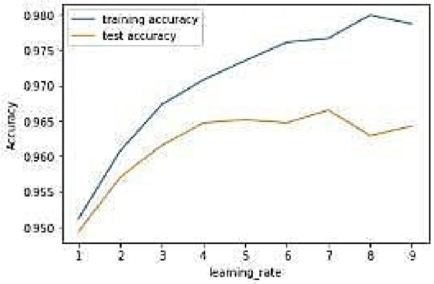


```
In [52]:
#computing the classification report of the model

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
```



```
In [53]:
training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10)
for n in depth:
    forest_test =  GradientBoostingClassifier(learning_rate = n*0.1)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("learning_rate")
plt.legend();
```



7.2 FEAUTRE 2

BUILDING FLASK APP

```
#importing required libraries


from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
```

```
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("pickle/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)
```

FLASK 2

```
import ipaddress

import re

import urllib.request

from bs4 import BeautifulSoup

import socket

import requests

from googlesearch import search

import whois

from datetime import date, datetime

import time

from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

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

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

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

```
    pass

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



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


self.features.append(self.NonStdPort())
```

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

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


# 1.UsingIp
```

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

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

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

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

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

'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

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

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


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


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

```
        return -1
    return 1

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

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

# 8.HTTPS
```

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

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

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

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

    # 11. NonStdPort
```

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

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

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

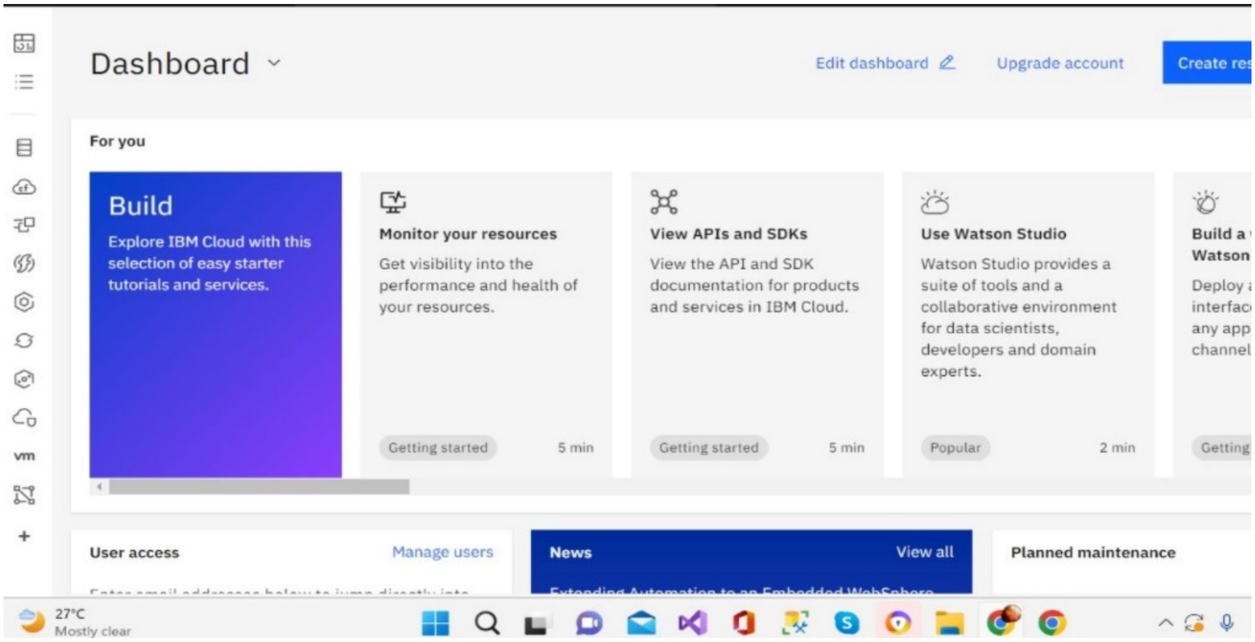
```
== 1:
                    success = success + 1
                i = i+1

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

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

            for iframe in self.soup.find_all('iframe', src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1
            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

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

        except:
            return -1

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

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

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

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

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


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

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


# 19. WebsiteForwarding
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
         return -1
    # 20. StatusBarCust
    def StatusBarCust(self):
        try:
            if re.findall("<script>.+onmouseover.+</script>",
self.response.text):
                return 1
            else:
                return -1
        except:
             return -1


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


    # 22. UsingPopupWindow
    def UsingPopupWindow(self):
        try:
```

```
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

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

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

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

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

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

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

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

```
self.domain})

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



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

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

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

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.co
m|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\.18
5\.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|17
5\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.23
2\.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|1
95\.16\.127\.102|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.6
4\.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\.8
6\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.23
1\.42', ip_address)
            if url_match:
                return -1
            elif ip_match:
                return -1
            return 1
        except:
            return 1


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

7.3 STORING THE APP IN IBM CLOUD STORAGE SERVICE



Explore IBM Cloud with this selection of easy starter tutorials and services.

Get visibility into the performance and health of your resources.

View the API and SDK documentation for products and services in IBM Cloud.

Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

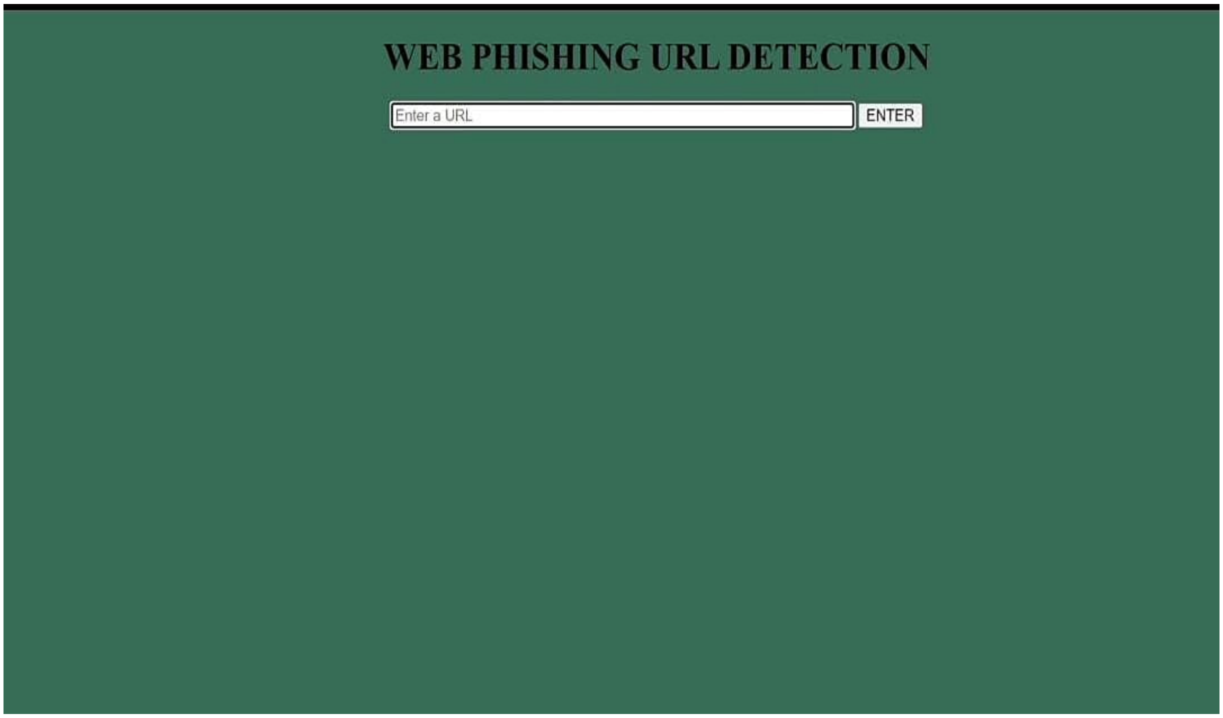
Deploy a interfac any app channel

INTEGRATE FLASK WITH SCORING ENDPOINTS

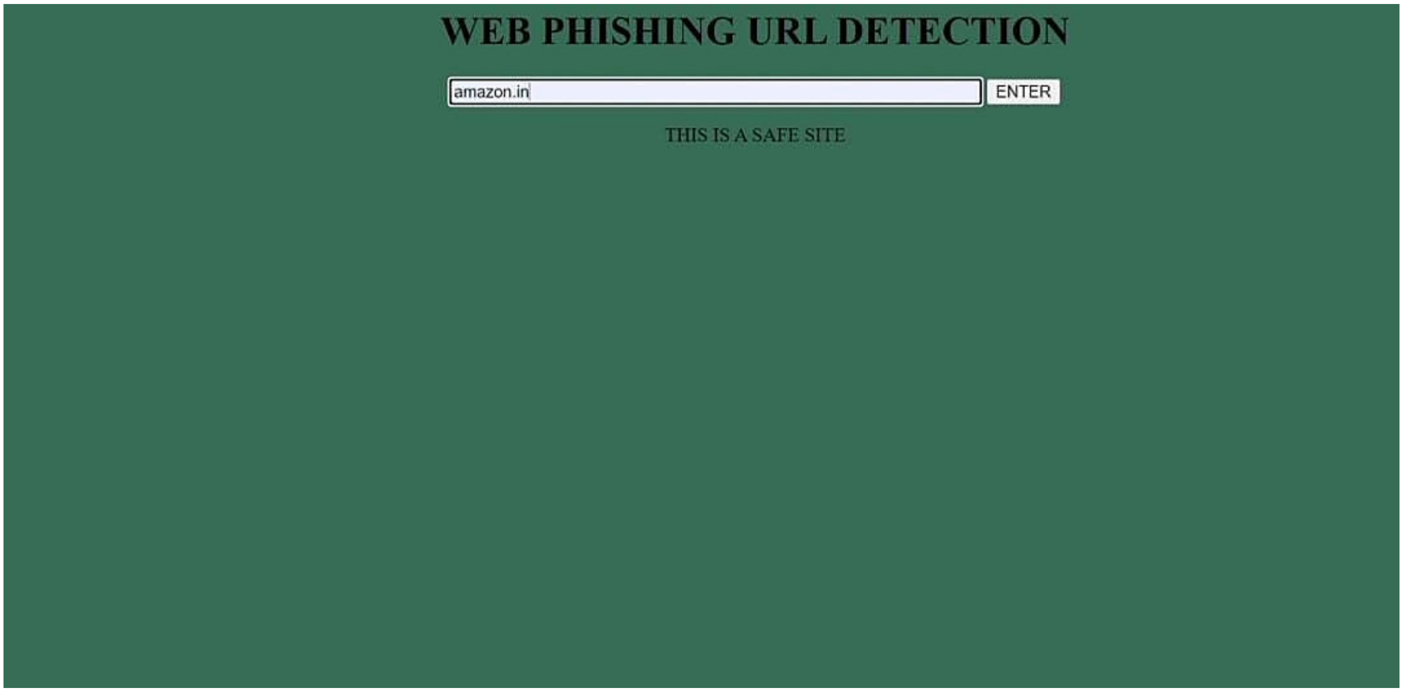
```
#importing required libraries


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



# NOTE: you must manually set API_KEY below using information
```

```
retrieved from your IBM Cloud account.
API_KEY = ""
token_response =
requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}



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

#Redirects to the page to give the user input URL.
@app.route('/')
def predict():
    return render_template('index.html',result="")

#Fetches the URL given by the URL and passes to inputScript
@app.route('/',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    url = request.form['url']
    checkprediction = inputScript.main(url)
    print(url)
    print(checkprediction)
    prediction = model.predict(X=checkprediction)
    requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments//predictions?version=2022-11-06', json=prediction,
        headers={'Authorization': 'Bearer ' + mltoken})
    print(prediction)
    output=prediction[0]
    print(output)
    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', result=pred,url=url)


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

TESTING WEBPAGE



WEB PHISHING URL DETECTION



WEB PHISHING URL DETECTION

amazon.in

THIS IS A SAFE SITE

CHAPTER 8

TESTING TABLE

8.1 TEST CASES

Model Performance Testing

S.No.	**Parameter**	**Values**	**Screenshot**
1.	Model Summary	Decision Tree ModelAccuracy – 95**%**	
2.	Accuracy	Training Accuracy -Test Accuracy -	

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

TABLE 8.2

USER ACCEPTANCE TESTING

Resolution	Severit y 1	Severit y 2	Severit y 3	Severit y 4	Subt otal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. High Level of accuracy while comparing to other algorithms and methodology
2. Fast in Classification Process
3. When it is built in banking sectors , our proposed system will identify the phishing websites and deny the request further more if it is a phishing website.

4. By learning this each one can gain an awareness on Phishing attacks
5. Preventing financial fraud and embezzlement
6. Prevention of cyber espionage
7. Prevention of fraud through financial transactions like wire transfers etc.
8. Protects your business. One of the most significant advantages of having Cybersecurity is it provides extensive protections for digital anomalies.

DISADVANTAGES

1. It is needed to be monitored continuously so the bandwidth is consumed more
2. It has huge number of features, so the classifying process is challenging part
3. The Web server has some delay due to Python-Flask Environment was implemented.
4. Day by day technology improves as well as negative impacts is also increased so as we must be updated to upcoming technologies.

CHAPTER 11

CONCLUSION

We discuss our large-scale system for automatically categorizing phishing runs in this design, which has a false positive rate with less than 0.1. In a fraction of the time, it takes a customized review procedure, our bracket system reviews millions of implicit phishing runner's responses. We reduce the amount of time that phishing runners can be active before we protect our druggies by automatically simplifying our blacklist with our classifier. Indeed, our blacklist strategy keeps us a step ahead of the phishers, thanks to a superb classifier and a robust system. Using the machine literacy method, we can only distinguish between phishing and legitimate URLs. In terms of the delicacy meter, this is what we obtained.