IBM NALAIYA THIRAN

PROJECT REPORT

BACHELOR OF ENGINEERING *in*

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY: CHENNAI 600 025

Team Leader : SAI NANDHAN AP

Team members :

1.SAI NANDHAN AP

2 SANJEEV KARTHICK K

3 YASHWANTH BT

4 MUHAMMED RAMEEZ M

November 2022

ABSTRACT

A web service is one of the most important Internet communications software services. Using fraudulent methods to get personal information is becoming increasingly widespread these days. However, it makes our lives easier, it leads to numerous security vulnerabilities to the Internet's private structure. Web phishing is just one of the many security risks that web services face. Phishing assaults are usually detected by experienced users however, security is a primary concern for system users who are unaware of such situations. Phishing is the act of portraying malicious web runners as genuine web runners to obtain sensitive information from the end-user. Phishing is currently regarded as one of the most dangerous threats to web security. Vicious Web sites significantly encourage Internet criminal activity and inhibit the growth of Web services. As a result, there has been a tremendous push to build a comprehensive solution to prevent users from accessing such websites. We suggest a literacy-based strategy to categorize Web sites into three categories: benign, spam, and malicious. Our technology merely examines the Uniform Resource Locator (URL) itself, not the content of Web pages. As a result, it removes run-time stillness and the risk of drug users being exposed to cyber surfer-based vulnerabilities. When compared to a blacklisting service, our approach performs better on generality and content since it uses learning techniques.

TABLE OF CONTENTS

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:

- 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.

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.

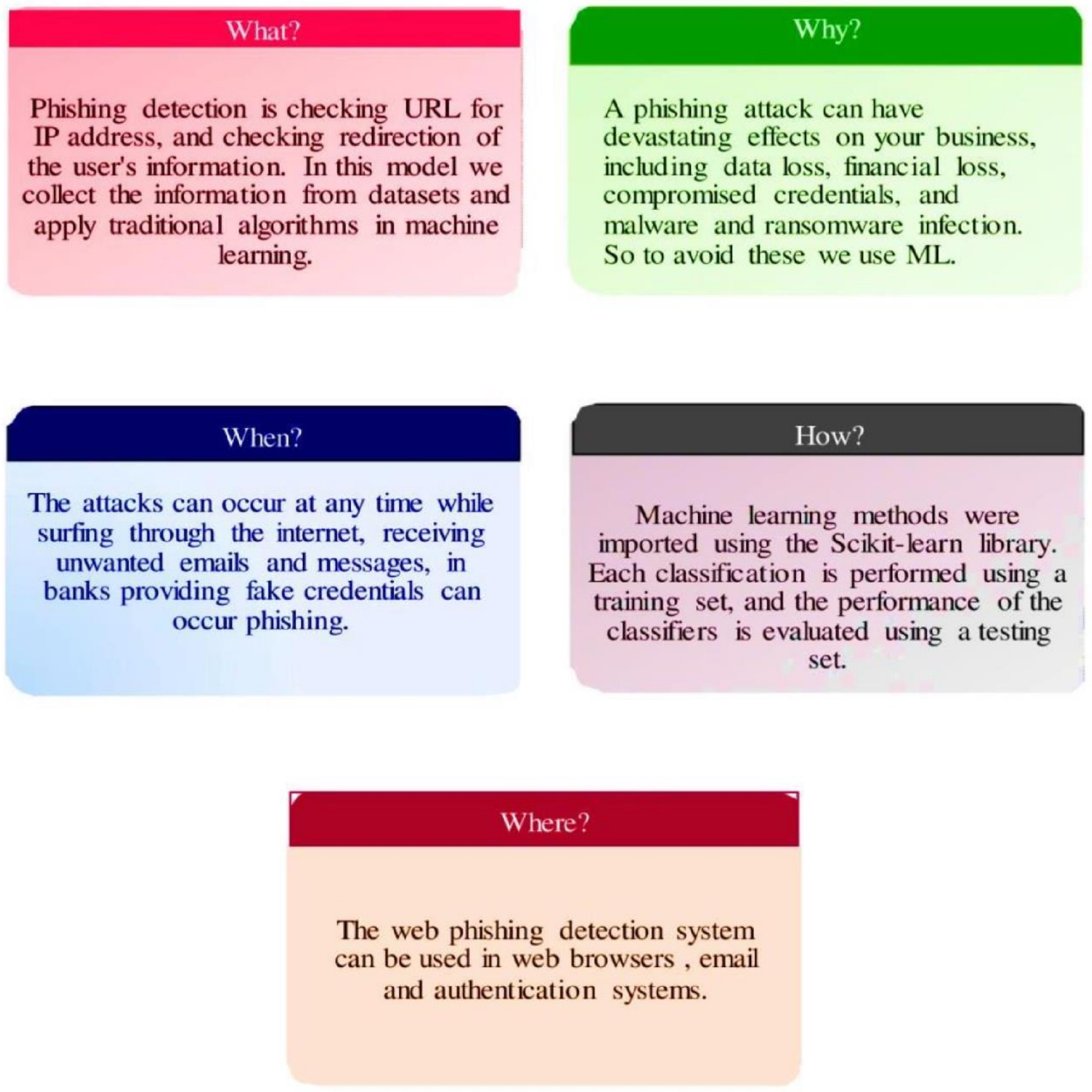
These critical issues have drawn many researchers to work on various approaches to improve detection accuracy of phishing attacks and to minimize false alarm rate. The inconsistent nature of attacks behaviors and continuously changing URL phish patterns require timely updating of the reference model. 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. To combat the ever evolving and complexity of phishing attacks and tactics, ML anti-phishing techniques are essential.

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.
- Quintin, Cooper (August 27, 2015). "New Spear Phishing Campaign Pretends to be EFF". EFF. Archived from the original on August 7, 2019. Retrieved November 29, 2016.

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.



What?

Phishing detection is checking URL for IP address, and checking redirection of the user's information. In this model we collect the information from datasets and apply traditional algorithms in machine learning.

Why?

A phishing attack can have devastating effects on your business, including data loss, financial loss, compromised credentials, and malware and ransomware infection. So to avoid these we use ML.

When?

The attacks can occur at any time while surfing through the internet, receiving unwanted emails and messages, in banks providing fake credentials can occur phishing.

How?

Machine learning methods were imported using the Scikit-learn library. Each classification is performed using a training set, and the performance of the classifiers is evaluated using a testing set.

Where?

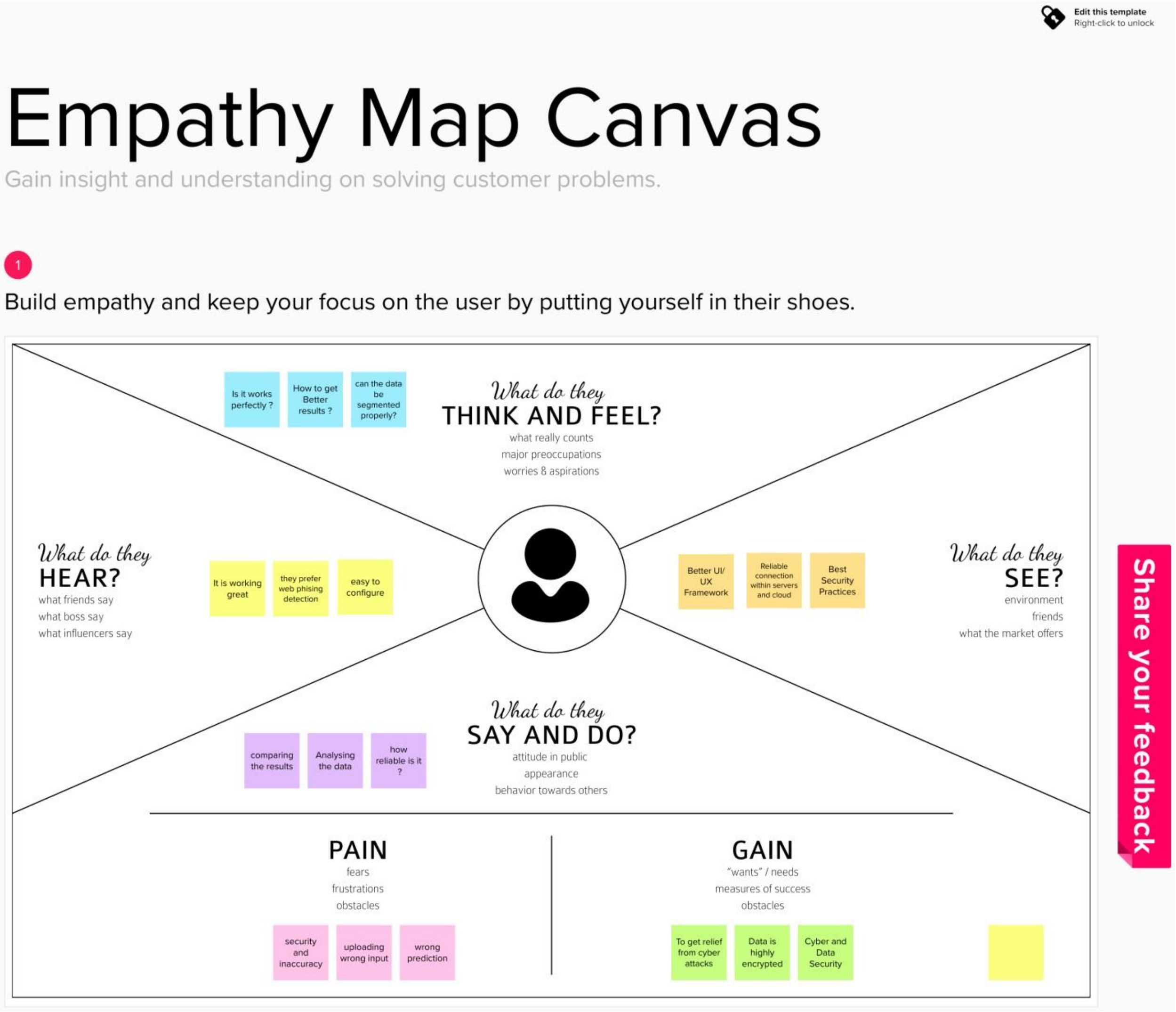
The web phishing detection system can be used in web browsers , email and authentication systems.

Figure 2.3 Problem Statement Definition

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

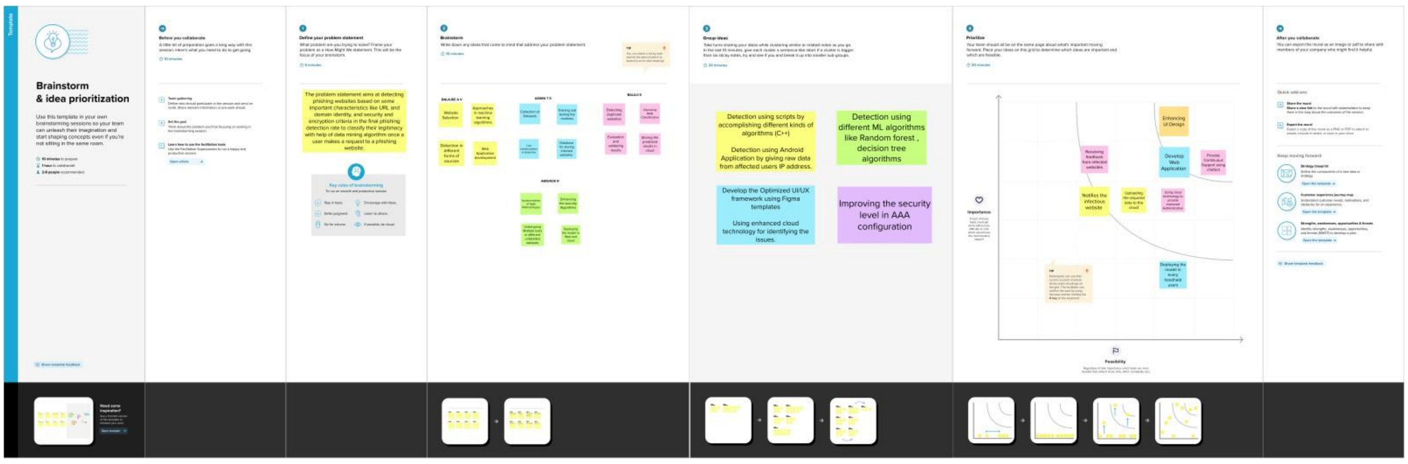
- security and inaccuracy
- uploading wrong input
- wrong prediction

GAIN
"wants" / needs
measures of success
obstacles

- To get relief from cyber attacks
- Data is highly encrypted
- Cyber and Data Security

Figure 3.1 Empathy Map Canvas

3.2 IDEATION & BRAINSTORMING



Brainstorm
& idea prioritization

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

The problem statement aims at detecting phishing websites based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate to classify their legitimacy with help of data mining algorithm once a user makes a request to a phishing website.

- Detection using scripts by accomplishing different kinds of algorithms (C++)
  Detection using Android Application by giving raw data from affected users IP address.
- Detection using different ML algorithms like Random forest , decision tree algorithms
- Develop the Optimized UI/UX framework using Figma templates
  Using enhanced cloud technology for identifying the issues.
- Improving the security level in AAA configuration

Figure3.2 Ideation & Brainstorming

3.3 PROPOSED SOLUTION

Table 3.3 Proposed Solution

Team ID	**PNT2022TMID52622**
Project Name	**Web Phishing Detection**
Date	**09.10.2022**

S.NO.	**PARAMETER**	**DESCRIPTION**
1.	Problem Statement (Problem to be solved)	Novel phishing approaches suffer low detection accuracy. The most common technique used is the blacklist-based method. It has become inefficient since registering a new domain has become easier. No comprehensive blacklist can ensure a perfect up-to-date database.
2.	Idea/ Solution Description	Our solution is to build an efficient and intelligent system to detect phishing sites by applying a machine learning algorithm that implements classification algorithms and techniques to extract the phishing datasets' criteria to classify their legitimacy.
3.	Novelty/ Uniqueness	We have carefully analysed and identified various factors that could be used to detect a phishing site. These factors fall under the categories of address bar-based features, domain-based features, and HTML & Javascriptbased features. Using these features we can identify a phishing site with high accuracy
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)	By generating leads we can improve our business model. By detecting the phishing sites, people won't access them which will reduce the revenue of malicious site owners.
6.	Scalability of Solution	This application can be accessed online without paying. It can be accessed via any browser of your choice. It can detect any site with high accuracy.

3.4 PROBLEM SOLUTION FIT

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

Purpose:

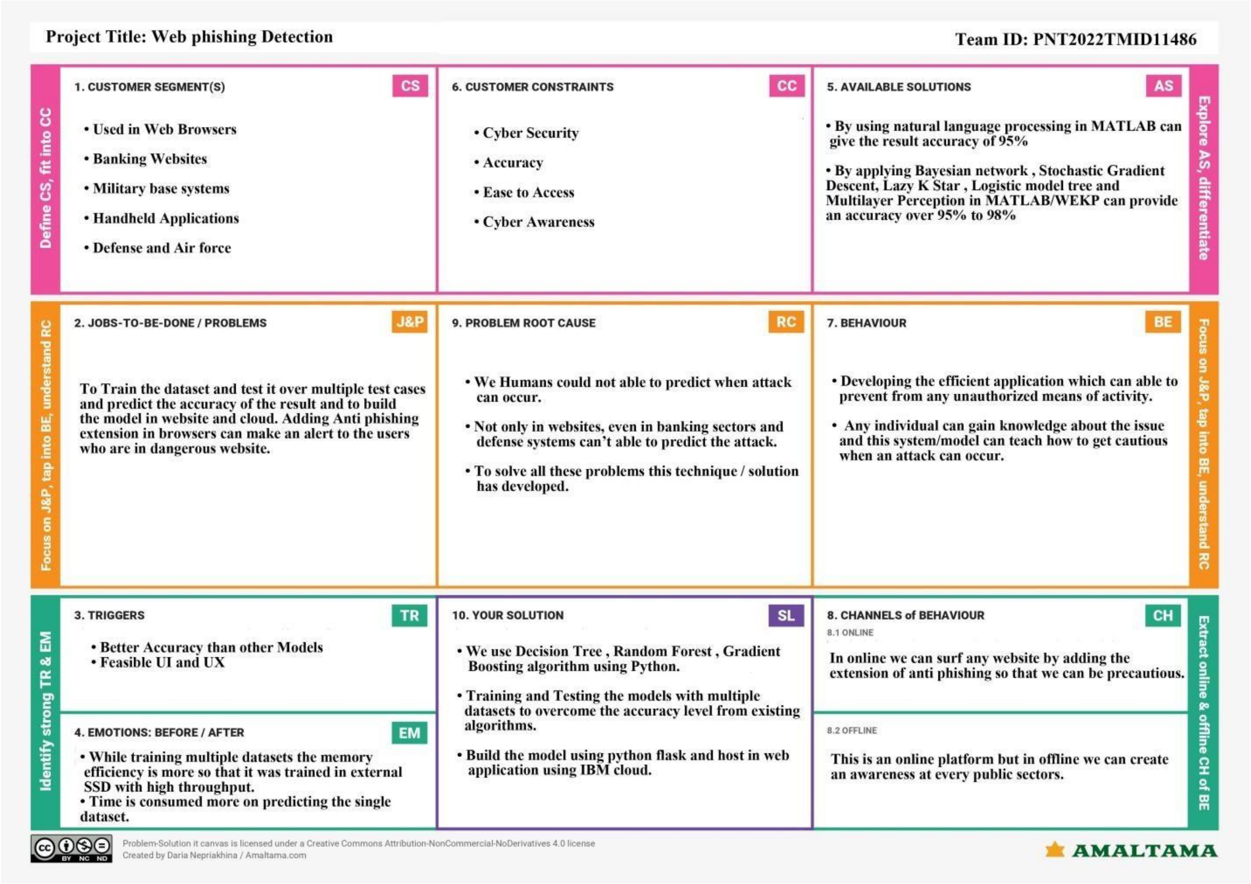
Θ Solve complex problems in a way that fits the state of your customers.

Θ Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior. Θ Sharpen your communication and marketing strategy with the right triggers and messaging.

Θ Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.

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

Template:


Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
Extract online & offline CH of BE
CS
CC
AS
J&P
RC
BE
TR
SL
CH
EM
AMALTAMA

Project Title: Web phishing Detection

Team ID: PNT2022TMID11486

1. CUSTOMER SEGMENT(S)

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

6. CUSTOMER CONSTRAINTS

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

5. AVAILABLE SOLUTIONS

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

2. JOBS-TO-BE-DONE / PROBLEMS

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

9. PROBLEM ROOT CAUSE

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

7. BEHAVIOUR

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

3. TRIGGERS

- Better Accuracy than other Models
- Feasible UI and UX

4. EMOTIONS: BEFORE / AFTER

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

10. YOUR SOLUTION

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

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

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

8.2 OFFLINE

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

Figure 3.4 Problem Solution Fit

CHAPTER 4 REQUIREMENT ANALYSIS

Solution Requirements (Functional & Non-functional)

Date	13 October 2022
Team ID	PNT2022TMID52622
Project Name	Project - Web Phishing Detection
Maximum Marks	4 Marks

TABLE 4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	- Registration through Form Registration through Gmail - Registration through LinkedIN
FR-2	User Confirmation	- Confirmation via Email - Confirmation via OTP
FR-3	User Authentication	Confirmation of Google Firebase

FR-4	User Security	Strong Passwords , 2FA and FIDO2.0 Webaucn
FR-5	User Performance	Usage of Legitimate websites, Optimize Network Traffic

TABLE 4.2 NON-FUNCTIONAL REQUIREMENTS

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

Date	13 October 2022
Team ID	PNT2022TMID11612
Project Name	Project – Web Phishing Detection
Maximum Marks	4 Marks

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. It shows how data enters and leaves the system, what changes the information, and where data is stored.

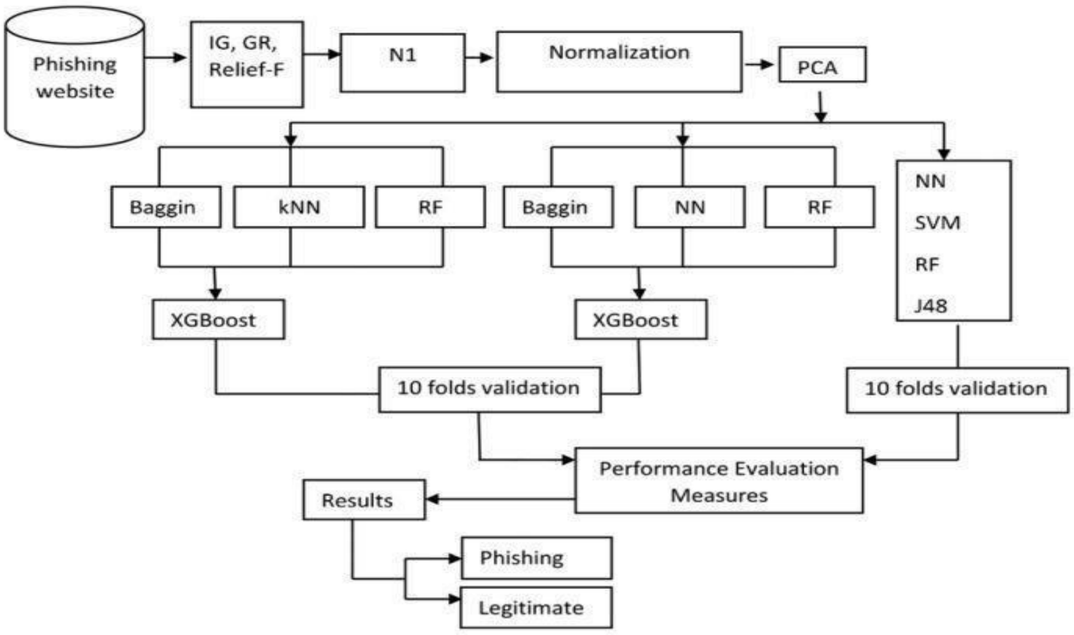


Figure 5.1 Data Flow Diagrams 5.2

SOLUTION & TECHNICAL ARCHITECTURE

Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	13 October 2022
Team ID	PNT2022TMID52622
Project Name	Project - Web Phishing Detection
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table 5.2.1 & table 5.2.2

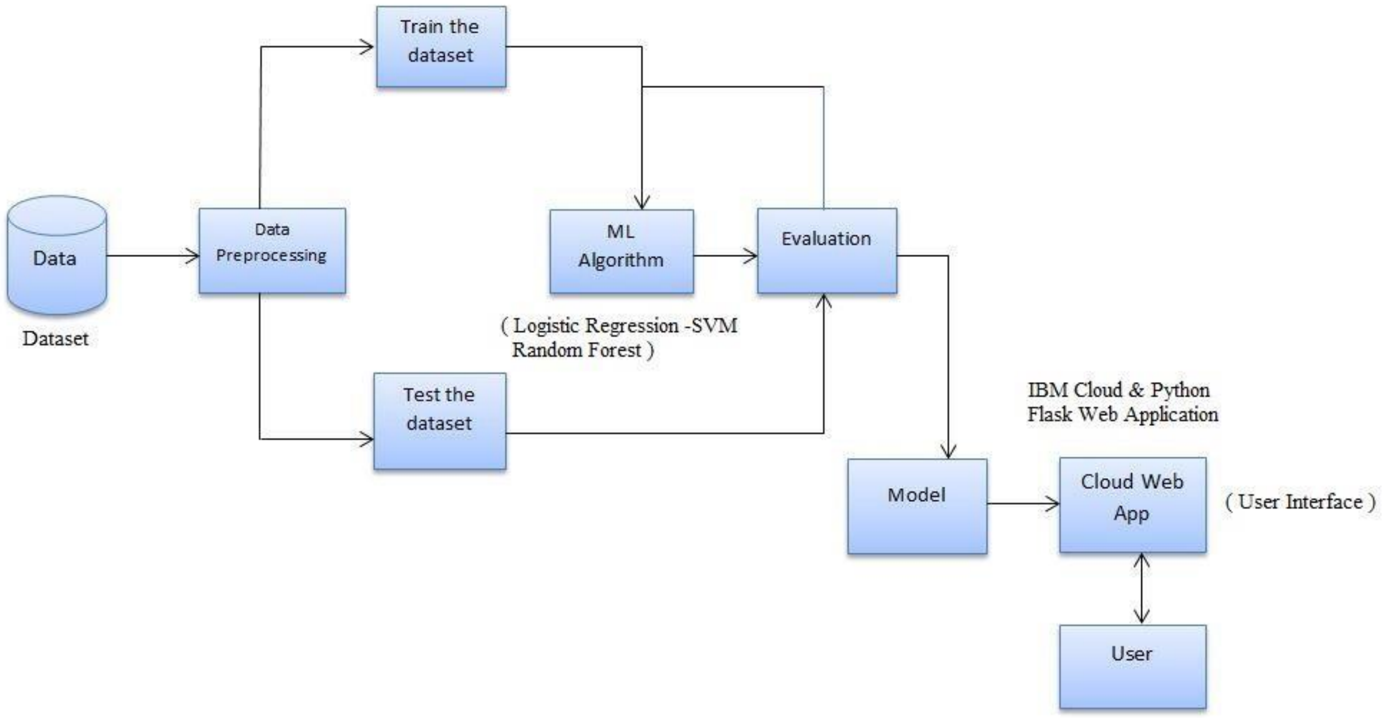


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web Application, Cloud UI	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Machine Learning Algorithms such as Gradient Boost, Random forest, Decision Tree, Logistic Regression and SVM. Python Flask Application for Web App	Java / Python
3.	Application Logic-2	IBM Watson Speech to Text technology enables fast and accurate speech transcription in multiple languages for a variety of use cases, including but not limited to customer self-service, agent assistance and speech analytics.	IBM Watson STT service
4.	Application Logic-3	The IBM Watson Assistant service combines machine learning, natural language understanding, and an integrated dialog editor to create conversation flows between your apps and your users.	IBM Watson Assistant
5.	Database	Stored Procedure (EXEC)	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Gophish is a powerful, open-source phishing framework that makes it easy to test your organization's exposure to phishing.	Machine Learning
2.	Security Implementations	In our prototype we use encryption techniques and security algorithms on web application	AES 256 , Cofense PDR
3.	Scalable Architecture	Scalability is high due to accuracy provided by the model and Responsive UI/UX	React Framework, jQuery, Bootstrap, Cloudfare
4.	Availability	Available at NLP, Spam Detection ,Blacklisting or Reporting, and machine learning techniques	Acunetix, Intruder, Ghost Phisher
5.	Performance	Deployed and Tested with multiple algorithms and this system gives greater accuracy and better performance than other.	Deep Learning

5.3 USER STORIES

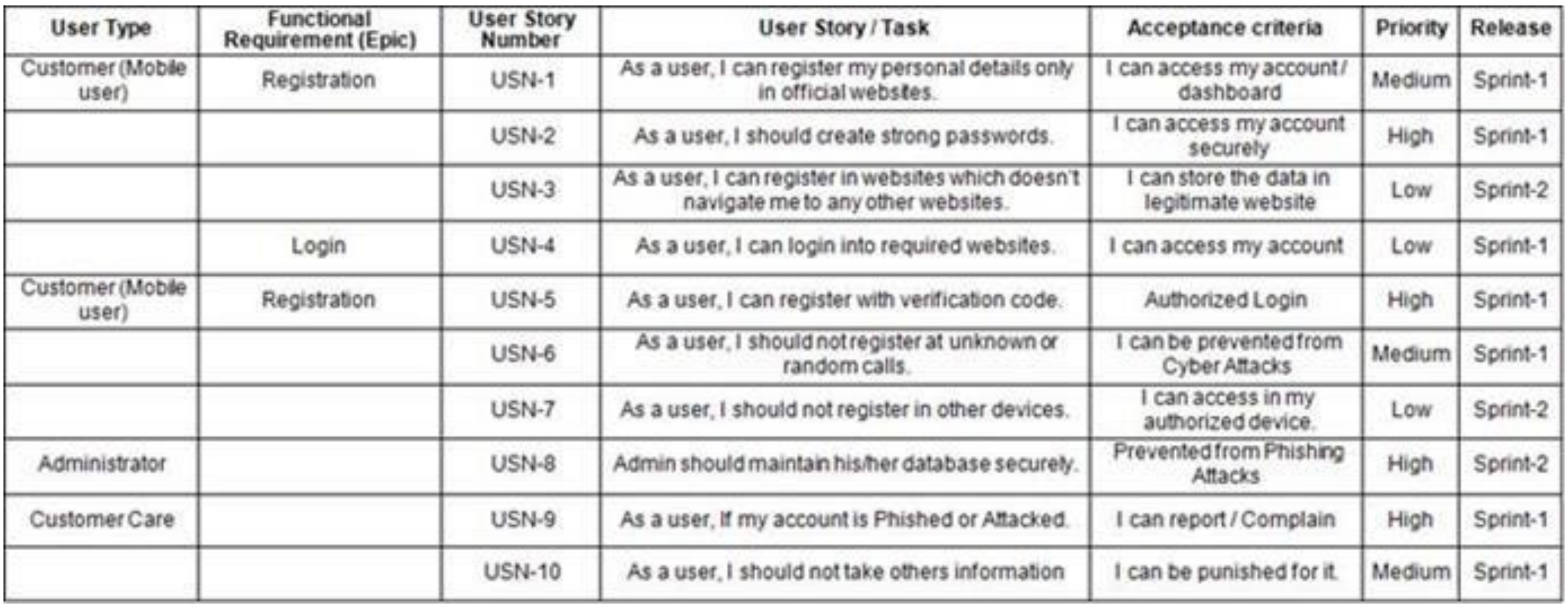
User stories are one of the core components of an agile program. They help provide a user-focused framework for daily work — which drives collaboration, creativity, and a better product overall.

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.

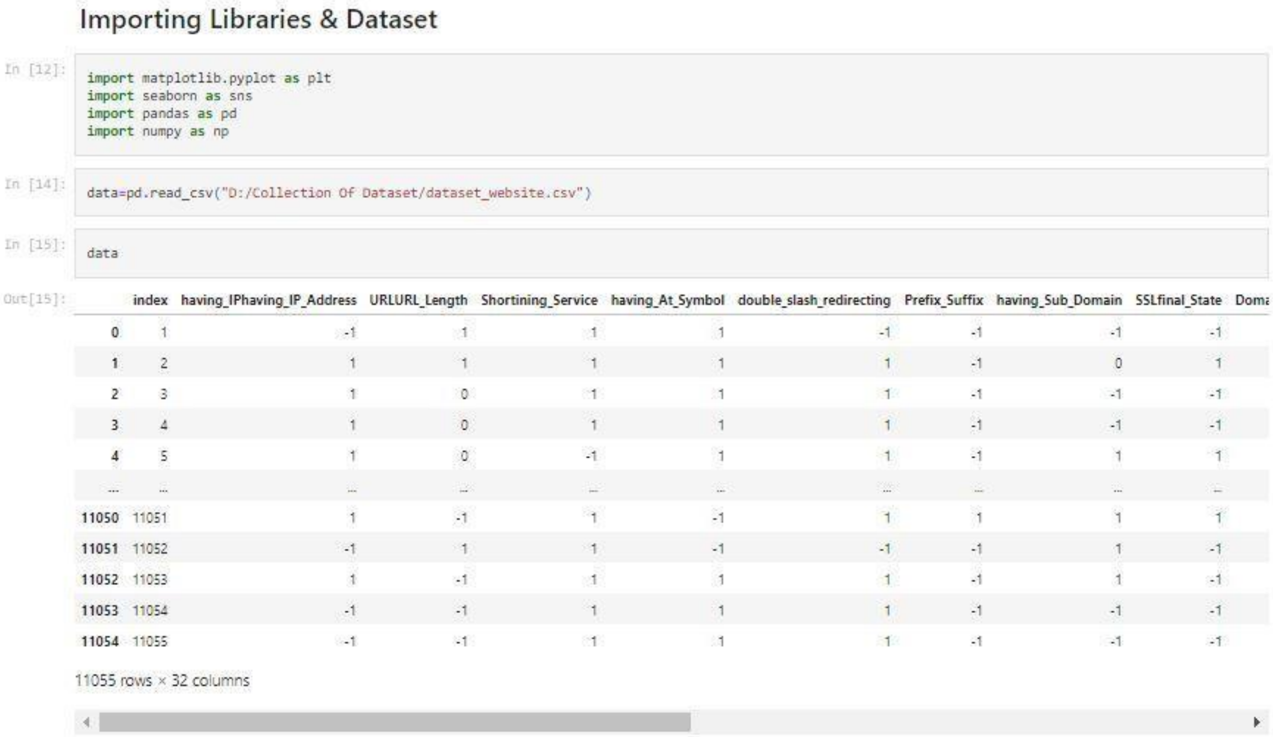
Table 5.3 User Stories



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register my personal details only in official websites.	I can access my account / dashboard	Medium	Sprint-1
		USN-2	As a user, I should create strong passwords.	I can access my account securely	High	Sprint-1
		USN-3	As a user, I can register in websites which doesn't navigate me to any other websites.	I can store the data in legitimate website	Low	Sprint-2
	Login	USN-4	As a user, I can login into required websites.	I can access my account	Low	Sprint-1
Customer (Mobile user)	Registration	USN-5	As a user, I can register with verification code.	Authorized Login	High	Sprint-1
		USN-6	As a user, I should not register at unknown or random calls.	I can be prevented from Cyber Attacks	Medium	Sprint-1
		USN-7	As a user, I should not register in other devices.	I can access in my authorized device.	Low	Sprint-2
Administrator		USN-8	Admin should maintain his/her database securely.	Prevented from Phishing Attacks	High	Sprint-2
Customer Care		USN-9	As a user, If my account is Phished or Attacked.	I can report / Complain	High	Sprint-1
		USN-10	As a user, I should not take others information	I can be punished for it.	Medium	Sprint-1

CHAPTER 6 CODING & SOLUTIONING 7.1 FEAUTRE 1 DATA PREPROCESSING & DATA VISUALIZATION

Data preprocessing is an **iterative process for the transformation of the raw data into understandable and useable forms**. Raw datasets are usually characterized by incompleteness, inconsistencies, lacking in behavior, and trends while containing errors . The preprocessing is essential to handle the missing values and address inconsistencies.



Importing Libraries & Dataset

```
In [12]:  import matplotlib.pyplot as plt
          import seaborn as sns
          import pandas as pd
          import numpy as np
```

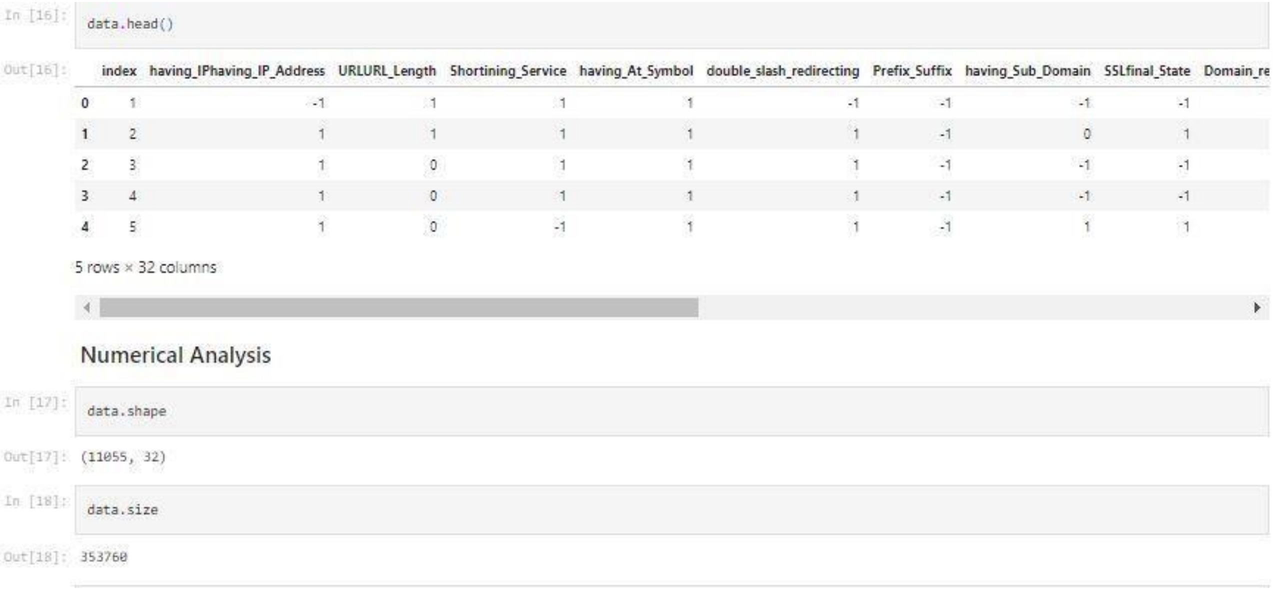
```
In [14]:  data=pd.read_csv("D:/Collection Of Dataset/dataset_website.csv")
```

```
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



```
In [16]:  data.head()
```

Out[16]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_re
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	

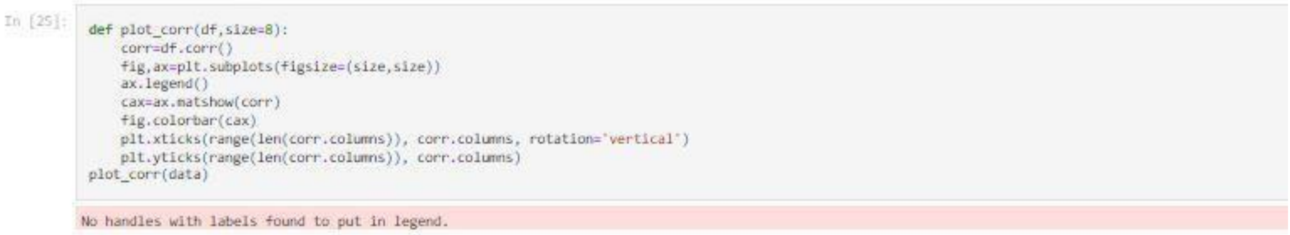
5 rows × 32 columns

Numerical Analysis

```
In [17]:  data.shape

Out[17]:  (11055, 32)
```

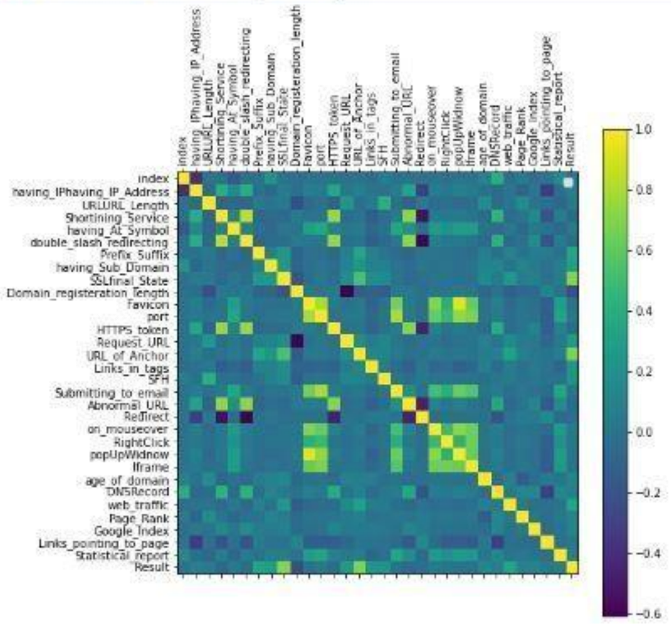
```
In [18]:  data.size

Out[18]:  353760
```

Data Visualization



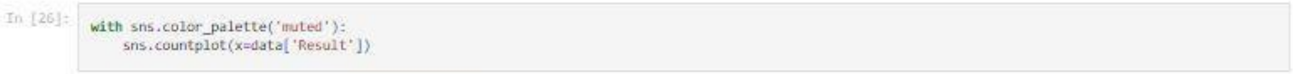
```
In [25]:
def plot_corr(df,size=8):
    corr=df.corr()
    fig,ax=plt.subplots(figsize=(size,size))
    ax.legend()
    cax=ax.matshow(corr)
    fig.colorbar(cax)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation='vertical')
    plt.yticks(range(len(corr.columns)), corr.columns)
plot_corr(data)
```

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

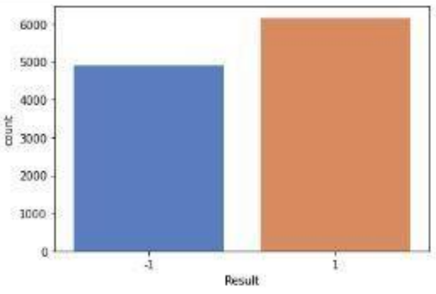


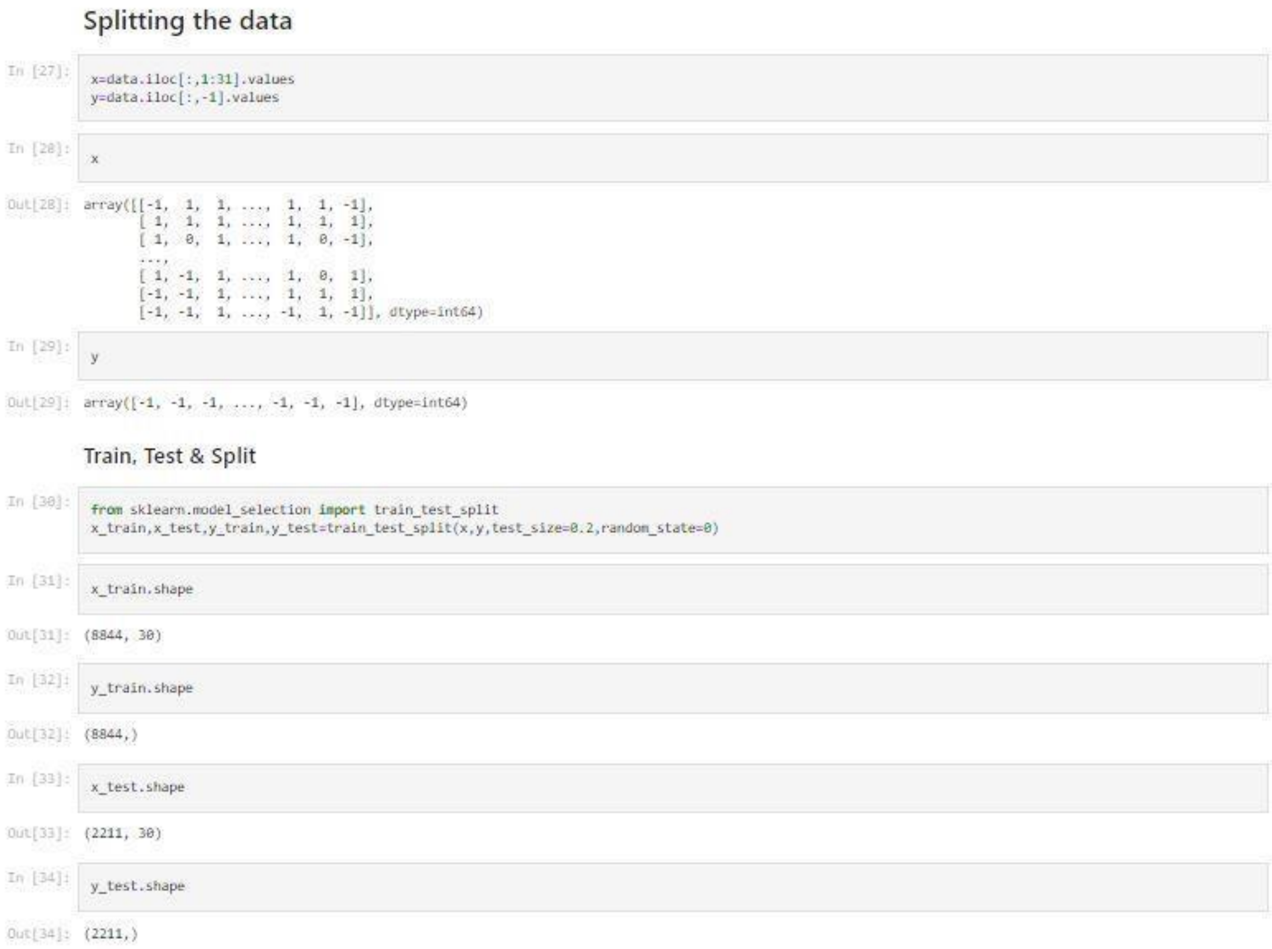
1. There are 11054 instances and 31 fearures in dataset.
2. Out of which 30 are independent features where as 1 is dependent feature.
3. Each feature is in int datatype, so there is no need to use LabelEncoder.
4. There is no outlier present in dataset.
5. There is no missing value in dataset

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

```
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

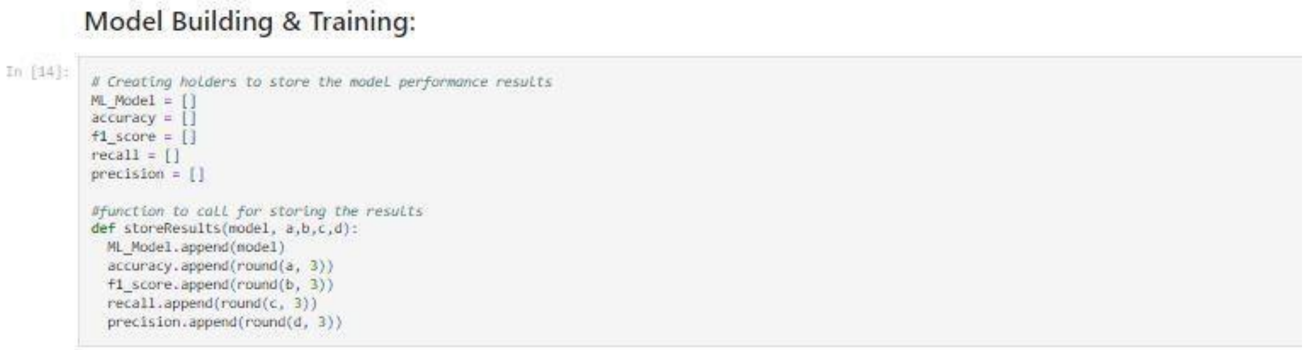
```
In [30]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

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,)
```

SELECTING APPROPRIATE MODEL

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

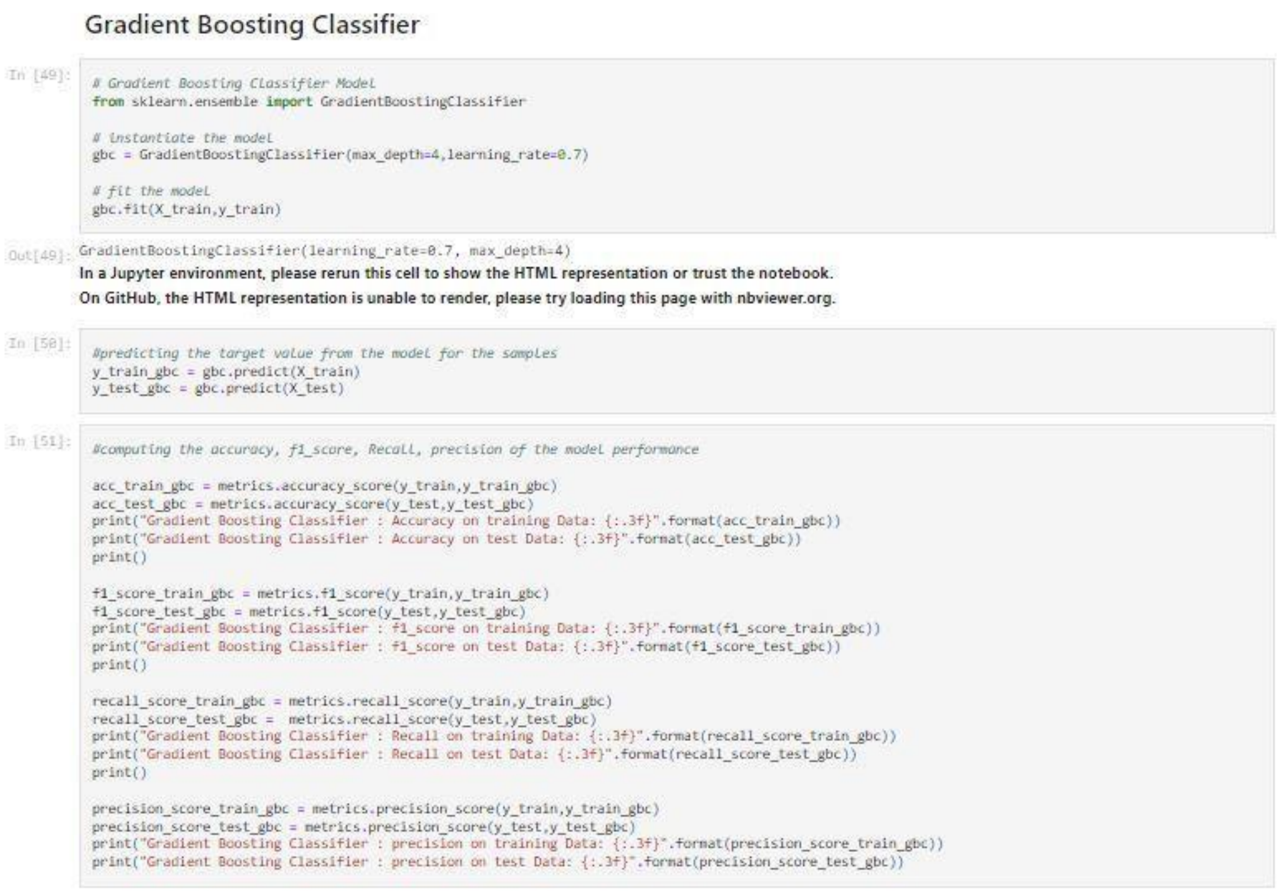
We’d reviewed a number of supervised machine learning models in class like Logistic Regression, K-Nearest Neighbors, Naive Bayes, Random Forest, and Gradient Boost.

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))
```



Gradient Boosting Classifier

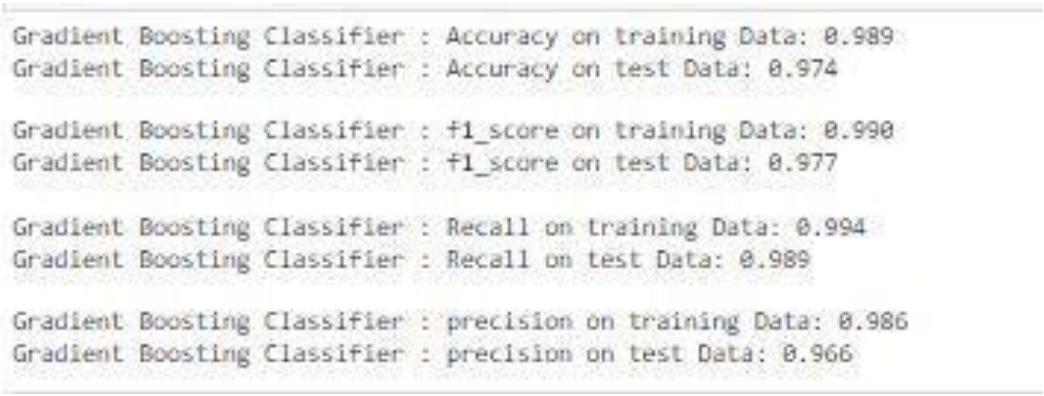
```
In [49]:
# Gradient Boosting Classifier Model
from sklearn.ensemble import GradientBoostingClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
```

```
Out[49]: GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [50]:
#predicting the target value from the model for the samples
y_train_gbc = gbc.predict(X_train)
y_test_gbc = gbc.predict(X_test)
```

```
In [51]:
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_gbc = metrics.accuracy_score(y_train,y_train_gbc)
acc_test_gbc = metrics.accuracy_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : Accuracy on training Data: {:.3f}".format(acc_train_gbc))
print("Gradient Boosting Classifier : Accuracy on test Data: {:.3f}".format(acc_test_gbc))
print()

f1_score_train_gbc = metrics.f1_score(y_train,y_train_gbc)
f1_score_test_gbc = metrics.f1_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_gbc))
print("Gradient Boosting Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_gbc))
print()

recall_score_train_gbc = metrics.recall_score(y_train,y_train_gbc)
recall_score_test_gbc =  metrics.recall_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : Recall on training Data: {:.3f}".format(recall_score_train_gbc))
print("Gradient Boosting Classifier : Recall on test Data: {:.3f}".format(recall_score_test_gbc))
print()

precision_score_train_gbc = metrics.precision_score(y_train,y_train_gbc)
precision_score_test_gbc = metrics.precision_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : precision on training Data: {:.3f}".format(precision_score_train_gbc))
print("Gradient Boosting Classifier : precision on test Data: {:.3f}".format(precision_score_test_gbc))
```



```
Gradient Boosting Classifier : Accuracy on training Data: 0.989
Gradient Boosting Classifier : Accuracy on test Data: 0.974

Gradient Boosting Classifier : f1_score on training Data: 0.990
Gradient Boosting Classifier : f1_score on test Data: 0.977

Gradient Boosting Classifier : Recall on training Data: 0.994
Gradient Boosting Classifier : Recall on test Data: 0.989

Gradient Boosting Classifier : precision on training Data: 0.986
Gradient Boosting Classifier : precision on test Data: 0.966
```

CLASSIFICATION REPORT OF THE MODEL:

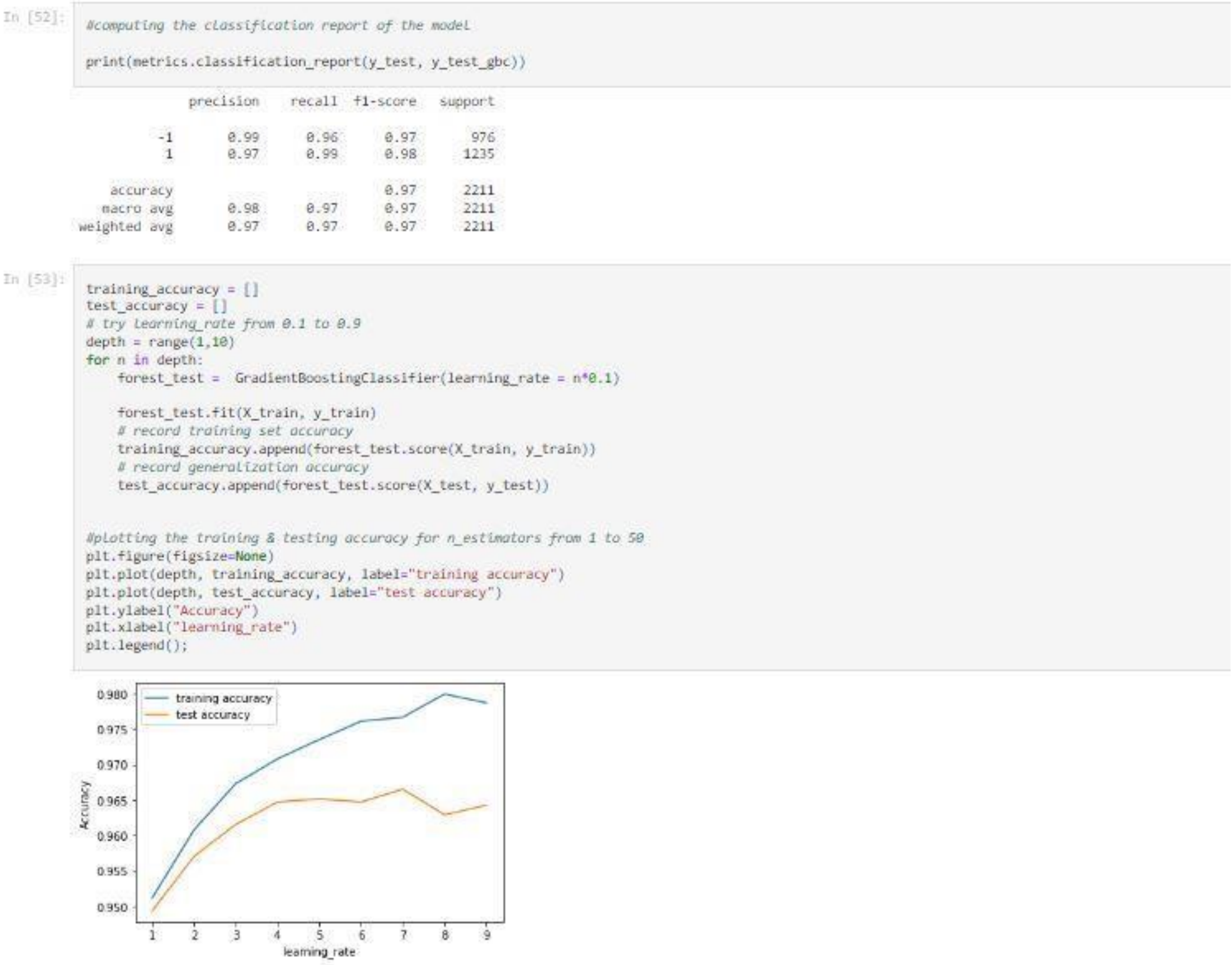
It is one of the performance evaluation metrics of a classification-based machine learning model. It displays your model's precision, recall, F1 score and support. It provides a better understanding of the overall performance of our trained model.

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.

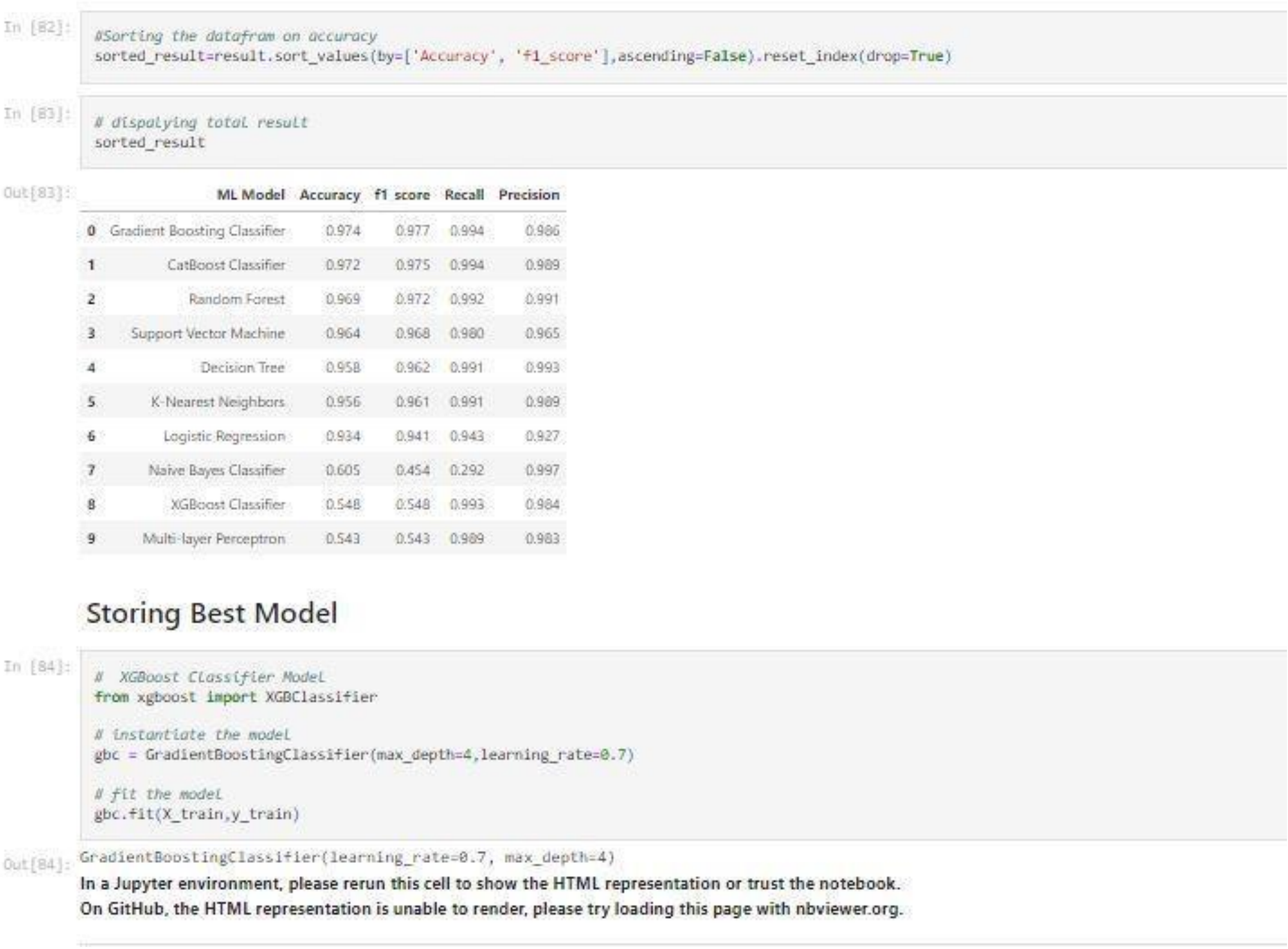
Support - Support is the number of actual occurrences of the class in the dataset. It doesn’t vary between models, it just diagnoses the performance evaluation process.



```
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();
```



In [82]:
```
#Sorting the datafram on accuracy
sorted_result=result.sort_values(by=['Accuracy', 'f1_score'],ascending=False).reset_index(drop=True)
```

In [83]:
```
# dispalying total result
sorted_result
```

Out[83]:

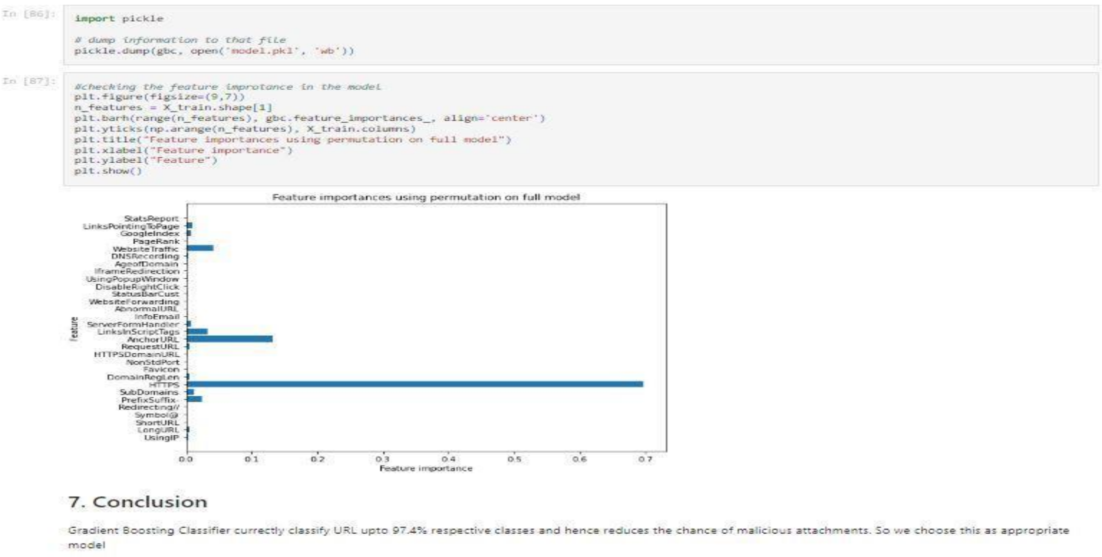
	ML Model	Accuracy	f1 score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

Storing Best Model

In [84]:
```
#  XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
```

Out[84]:
```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



In [86]:
```
import pickle

# dump information to that file
pickle.dump(gbc, open('model.pkl', 'wb'))
```

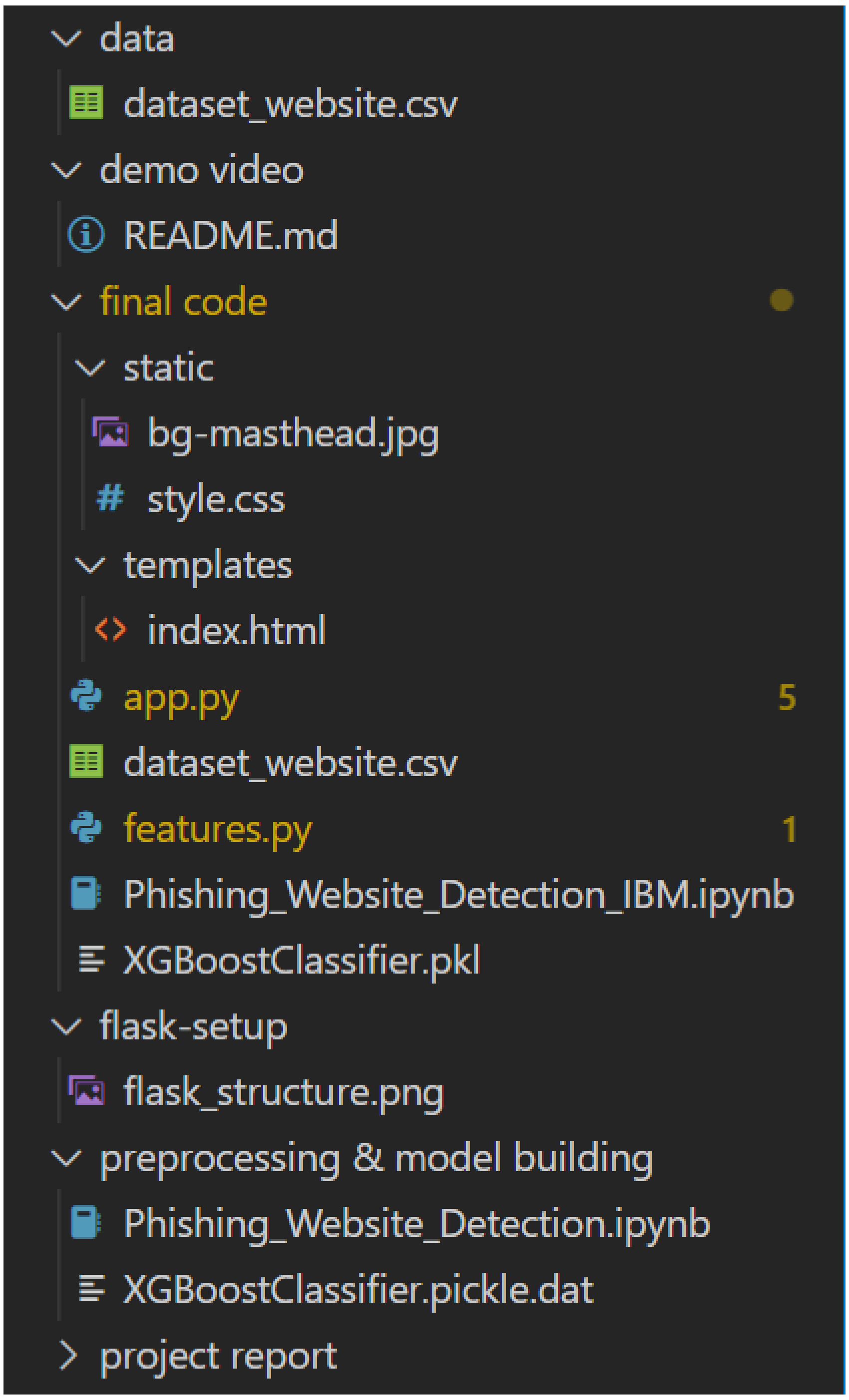
In [87]:
```
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), gbc.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.title("Feature importances using permutation on full model")
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```

7. Conclusion

Gradient Boosting Classifier currectly classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments. So we choose this as appropriate model

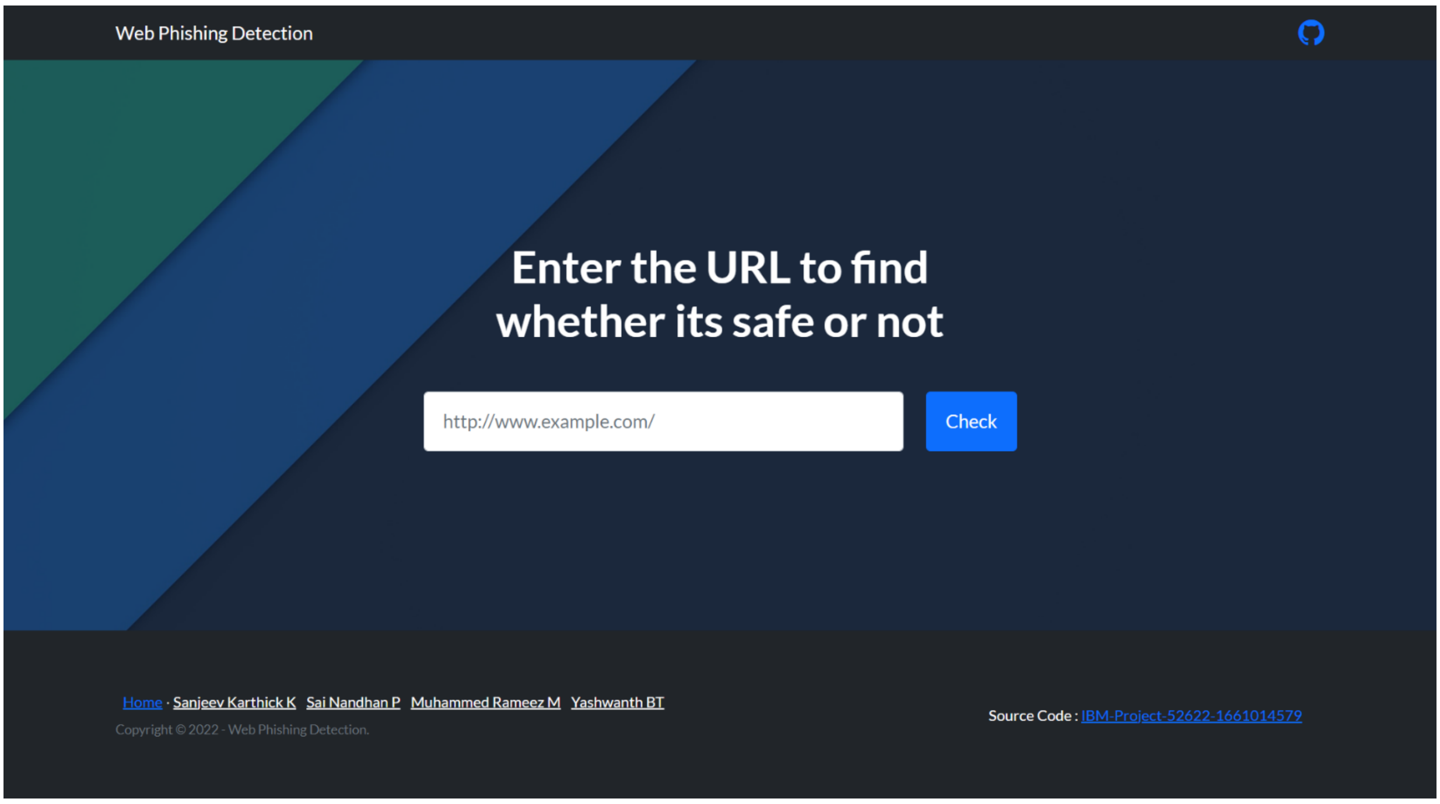
7.2 FEAUTRE 2

Flask Setup:



```
data
  dataset_website.csv
demo video
  README.md
final code
  static
    bg-masthead.jpg
    style.css
  templates
    index.html
  app.py                                 5
  dataset_website.csv
  features.py                            1
  Phishing_Website_Detection_IBM.ipynb
  XGBoostClassifier.pkl
flask-setup
  flask_structure.png
preprocessing & model building
  Phishing_Website_Detection.ipynb
  XGBoostClassifier.pickle.dat
project report
```

PUBLISHING AND TESTING WEBPAGE



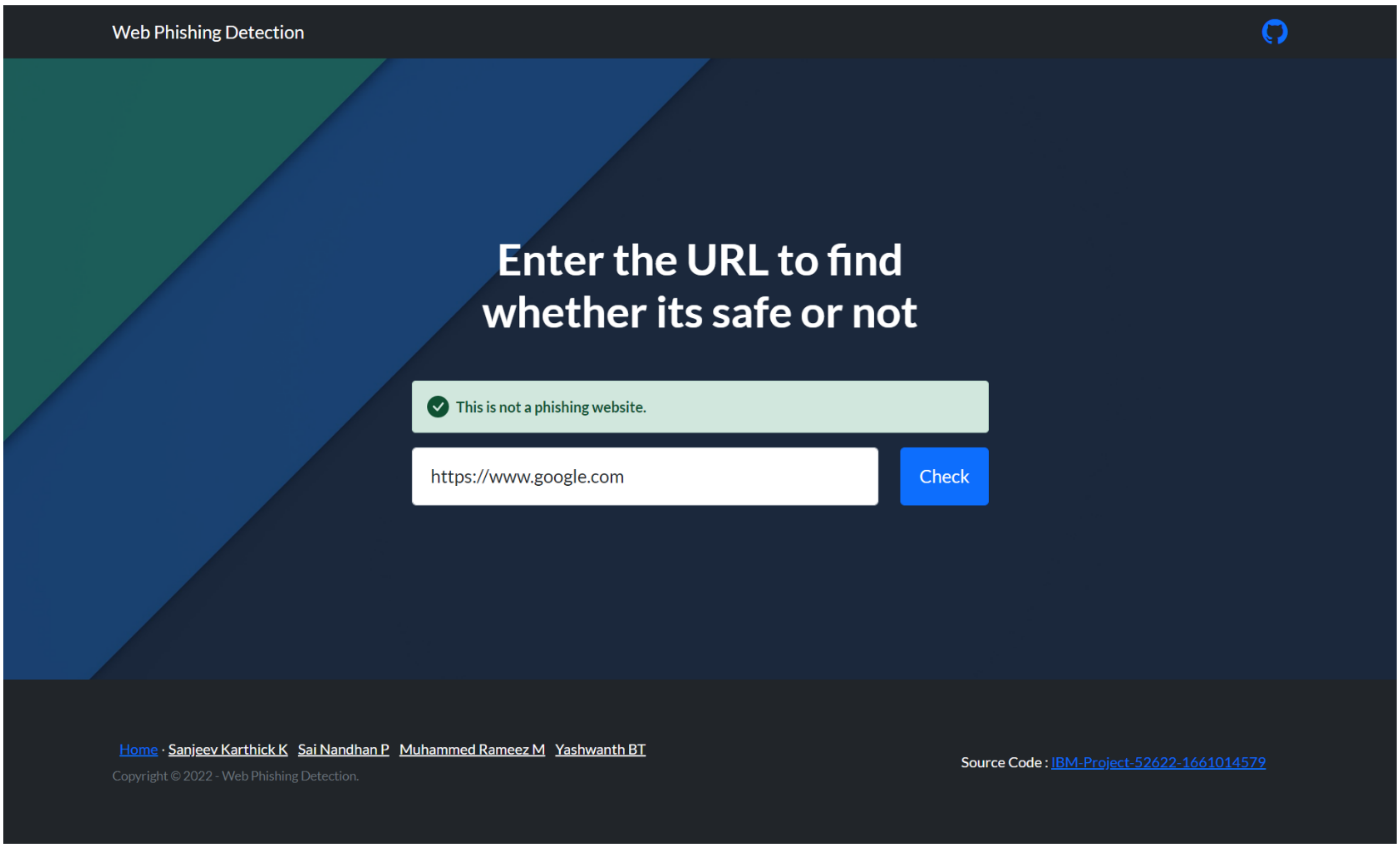
Enter the URL to find whether its safe or not

http://www.example.com/

Home · Sanjeev Karthick K Sai Nandhan P Muhammed Rameez M Yashwanth BT

Copyright © 2022 - Web Phishing Detection.

Source Code : IBM-Project-52622-1661014579



Enter the URL to find whether its safe or not

This is not a phishing website.

https://www.google.com

Home · Sanjeev Karthick K Sai Nandhan P Muhammed Rameez M Yashwanth BT

Copyright © 2022 - Web Phishing Detection.

Source Code : IBM-Project-52622-1661014579

CHAPTER 7 PERFORMANCE METRICS

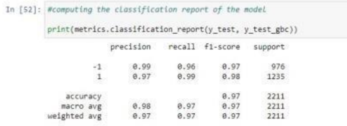
Project Development Phase Model Performance Test

Date	13 November 2022
Team ID	PNT2022TMID52622
Project Name	Project – Web Phishing Detection
Maximum Marks	10 Marks

Model Performance Testing:

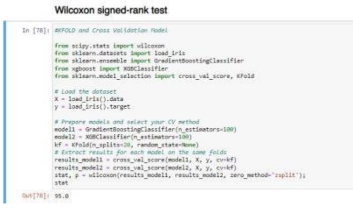
Project team shall fill the following information in model performance testing template.

<table>
<tr><th>S.No.</th><th>Parameter</th><th>Values</th><th>Screenshot</th></tr>
<tr><td>1.</td><td>Metrics</td><td><b>Classification Model:</b>
<b>Gradient Boosting Classification</b>
Accuray Score- 97.4%</td><td>



```
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
```

</td></tr>
<tr><td>2.</td><td>Tune the Model</td><td>Hyperparameter Tuning - 97%
Validation Method – KFOLD & Cross Validation Method</td><td>



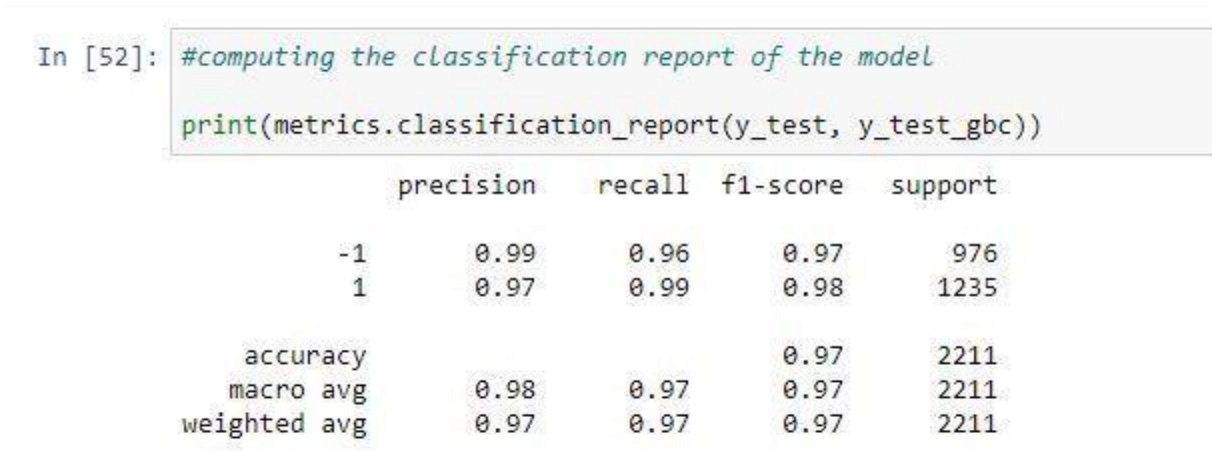
Wilcoxon signed-rank test

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

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

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

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

Out[78]: 95.0
```

</td></tr>
</table>

1. METRICS:

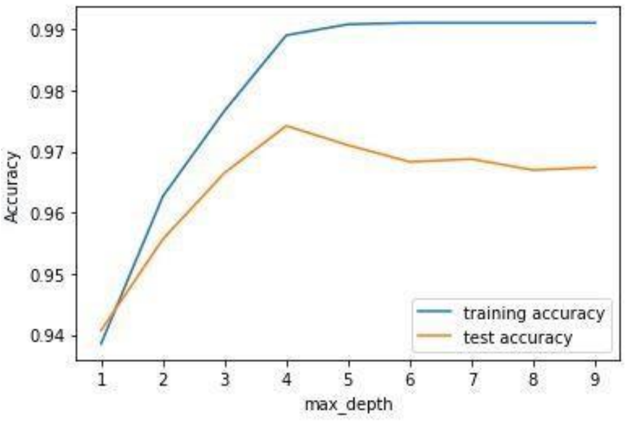
CLASSIFICATION REPORT:

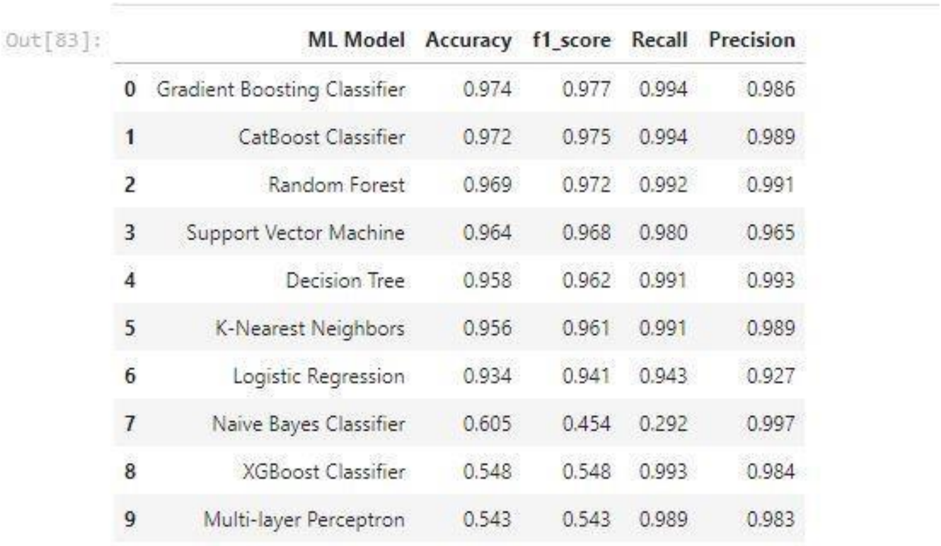


```
In [52]: #computing the classification report of the model

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

```
              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

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

PERFORMANCE :



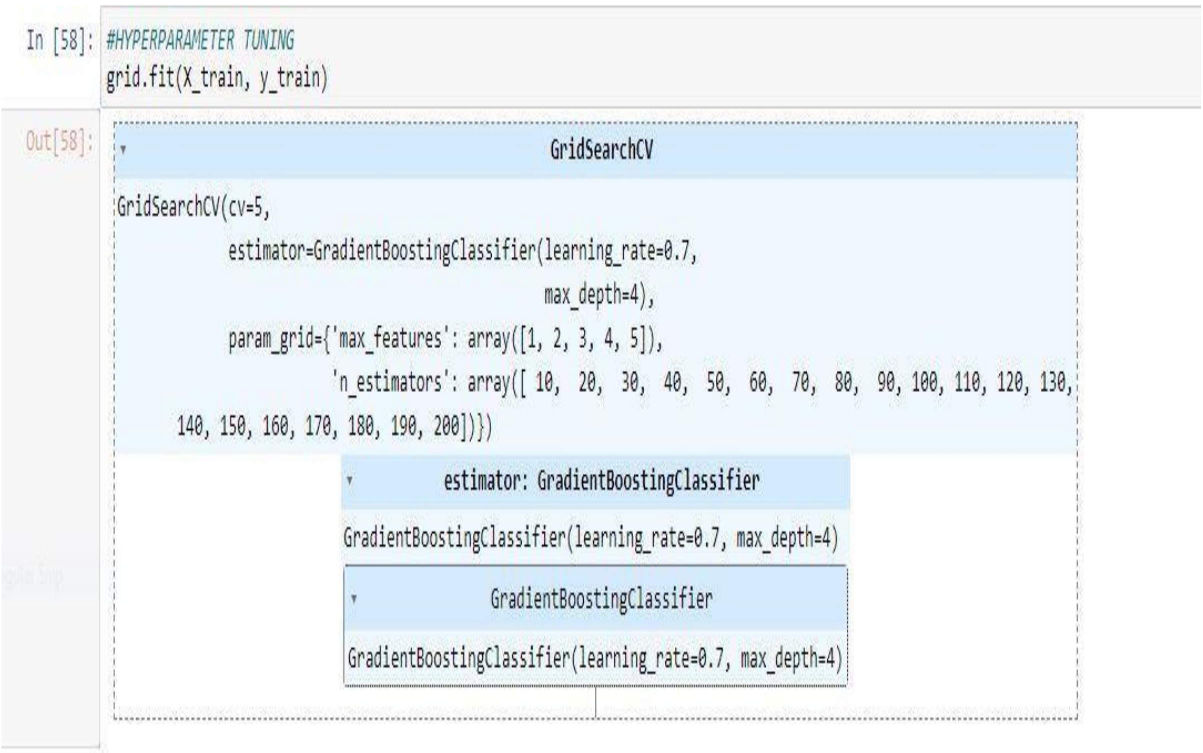


Out[83]:

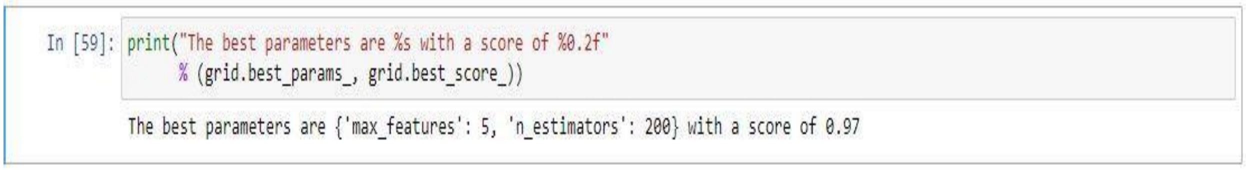
	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956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

2. TUNE THE MODEL – HYPERPARAMETER TUNING

Hyperparameter tuning is a hit and trial method where every combination of hyperparameters is tested and evaluated, and it selects the best model as the final model. Some scikit-learn APIs like GridSearchCV and RandomizedSearchCV are used to perform hyper parameter tuning.



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

Out[58]: GridSearchCV
GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                  max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})
estimator: GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```



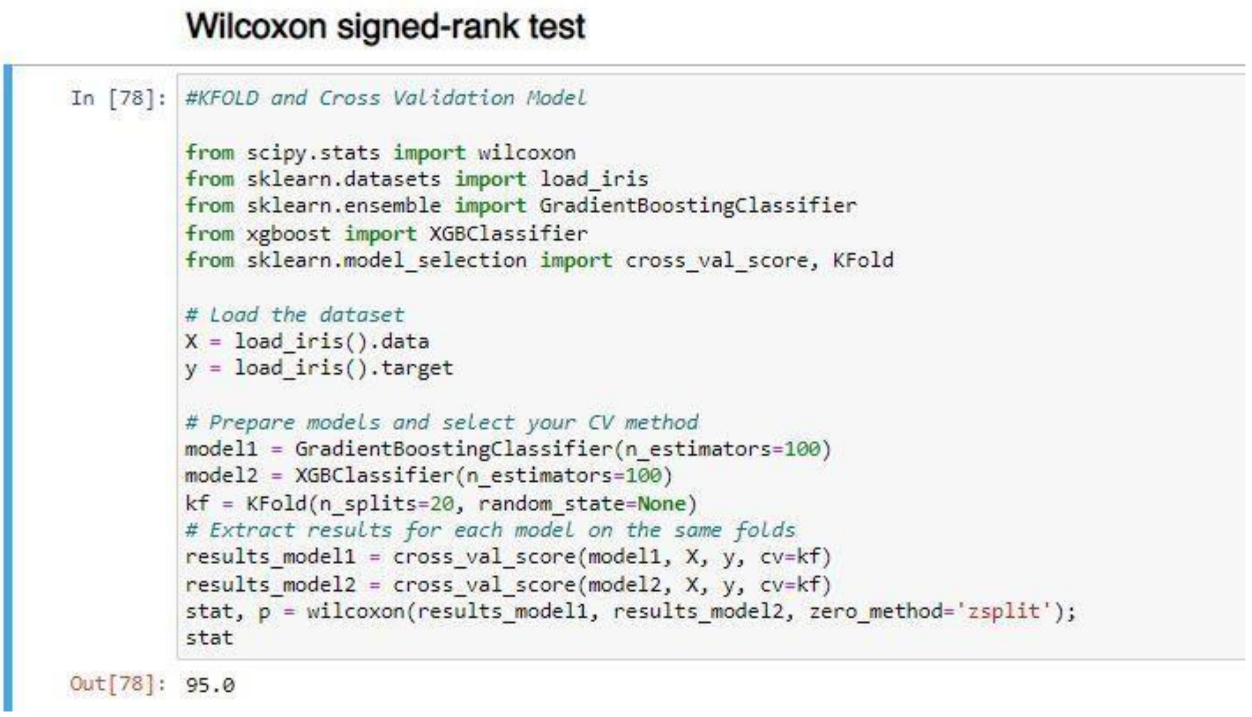
```
In [59]: print("The best parameters are %s with a score of %0.2f"
               % (grid.best_params_, grid.best_score_))

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

VALIDATION METHODS: KFOLD & Cross Folding

The validation set is used to evaluate a given model, but this is for frequent evaluation. We, as machine learning engineers, use this data to fine-tune the model hyperparameters. Hence the model occasionally *sees* this data, but never does it "*Learn*" from this. We use the validation set results, and update higher level hyperparameters. So the validation set affects a model, but only indirectly.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.



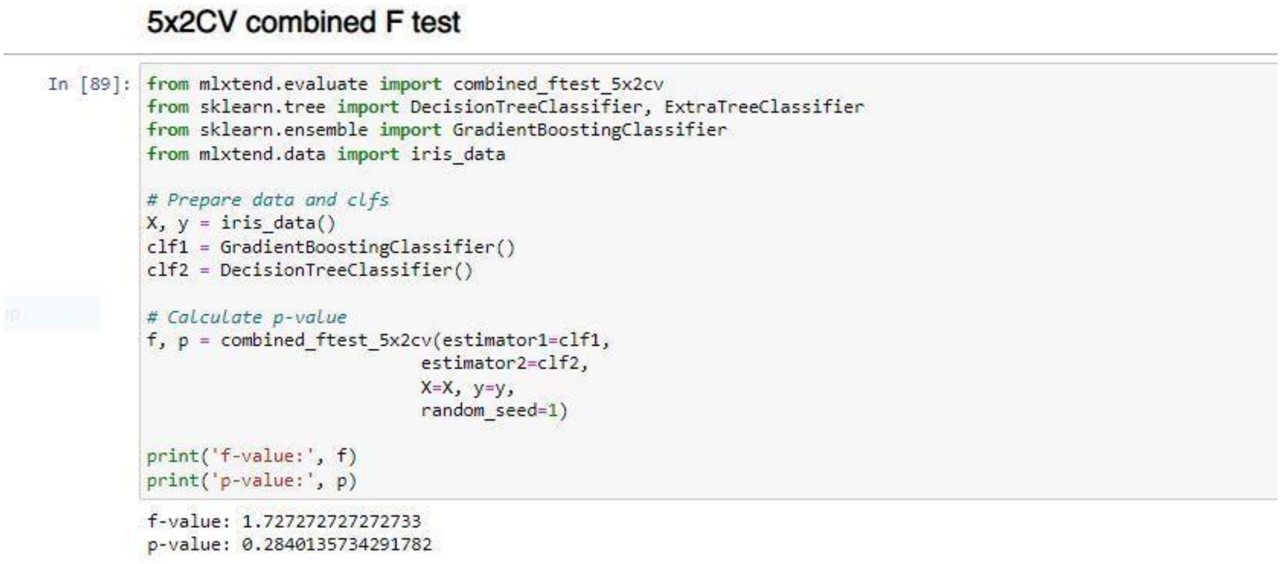
Wilcoxon signed-rank test

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

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

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

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

Out[78]: 95.0
```

The **Wilcoxon Signed-Rank Test** is a statistical test used to determine if 2 measurements from a single group are significantly different from each other on your variable of interest. Your variable of interest should be continuous and your group randomly sampled to meet the assumptions of this test.

The 5x2cv combined **F test** is a procedure for comparing the performance of two models (classifiers or regressors) that was proposed by Alpaydin as a more robust alternative to Dietterich's 5x2cv paired t-test procedure

.



5x2CV combined F test

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

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

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

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

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

CHAPTER 8

ADVANTAGES & DISADVANTAGES

ADVANTAGES

- High Level of accuracy while comparing to other algorithms and methodology
- Fast in Classification Process
- 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.
- By learning this each one can gain an awareness on Phishing attacks
- Preventing financial fraud and embezzlement
- Prevention of cyber espionage
- Prevention of fraud through financial transactions like wire transfers etc.
- Protects your business. One of the most significant advantages of having Cybersecurity is it provides extensive protections for digital anomalies.

DISADVANTAGES

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

CHAPTER 9 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.

CHAPTER 10

FUTURE SCOPE

Although the use of URL lexical features alone has been shown to result in high accuracy (97%), phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach .

For future enhancements, we intend to build the phishing detection system as a scalable extension and an anti-phish search engine which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

CHAPTER 11

APPENDIX

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-52622-1661014579