TEAM ID	**:**	**PNT2022TMID44694**	
TEAM MEMBERS	**:**	**DHIVYA S**	**(732519104004)**
		ABITHA S	**(732519104002)**
		VASUNTHARA R	**(732519104030)**
		PRIYATHARSHINI G	**(732519104703)**
DEPARTMENT	**:**	**CSE**	
COLLEGE NAME	**:**	**SHREE VENKATESHWARA HI-TECH ENGINEERING COLLEGE, GOBI**	

WEB PHISHING DETECTION

USING APPLIED DATA SCIENCE

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. The criminals, who want to obtain sensitive data, first create unauthorized replicas of a real website and e-mail. The e-mail will be created using logos and slogans of a legitimate company. The nature of website creation is one of the reasons that the Internet has grown so rapidly as a communication medium. Phisher then send the "spoofed" e-mails to as many people as possible in an attempt to lure them into the scheme. When these e-mails are opened or when a link in the mail is clicked, the consumers are redirected to a spoofed website, appearing to be from the legitimate entity. We discuss the methods used for detection of phishing Web sites based on url importance properties.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Internet constitutes a tremendous change in today's world because of its versatility. By utilising the sophisticated infrastructure of the internet, people can do transactions such as shopping, banking etc. whenever and wherever they want. The internet has many advantages, at the same time; it also has its own set of security and privacy problems. By using the anonymous and independent infrastructure of the internet, attackers create a prominent platform for cyber-attacks, such as phishing, malware distribution, privacy disclosure etc., which causes severe threats to the end-users of the internet.

Phishing is the most widespread and pernicious cyber attack , which mostly targets the human rather than the computer by exploiting their vulnerabilities. According to Anti-Phishing Working Group (APWG), phishing is a criminal mechanism employing both social engineering and technical tricks to steal user's identity data and financial account credentials by disguising as a trusted one. To lure the end-users, attackers use malicious websites and e-mails by posing themselves as a trusted one. The supreme goal of phishing is to abduct confidential data such as user name, password, bank details, credit card details etc. Attackers perform phishing for many reasons – to gain benefits financially, to steal personal information, to ruin the reputation of the organizations and sometimes just to get fame .

The term ‘phishing’ is coined in the mid-1990s and is from the term ‘fishing’ because it involves trying to outwit someone into a trap. The history of phishing scams can be traced back to the beginning of the 1990s via America Online (AOL), which was carried out by generating random credit card numbers to make fake AOL accounts . Later, attackers turned it into a million-dollar growth business, by impersonating many organizations such as banks, credit card companies, online payment service providers (e.g. InstaReM), and social media websites (e.g. Face book). Even Internet giants such as Google and Face book were scammed out of more than $100 million between 2013 and 2015 through email phishing . Recently in 2020, Texas school district lost $2.3 million in phishing raid and the Federal Bureau of Investigation is currently investigating on it.

1.2 PURPOSE

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

2. LITERATURE SURVEY

The purpose or goal behind phishing is data, money or personal information stealing through the fake website. The best strategy for avoiding the contact with the phishing web site is to detect real time malicious URL. Phishing websites can be determined on the basis of their domains. They usually are related to URL which needs to be registered (low-level domain and upper-level domain, path, query). Recently acquired status of intra-URL relationship is used to evaluate it using distinctive properties extracted from words that compose a URL based on query data from various search engines such as Google and Yahoo. These properties are further led to the machine-learning based classification for the identification of phishing URLs from a real dataset. This paper focus on real time URL phishing against phishing content by using phish - STORM. For this a few relationship between the register domain rest of the URL are consider also intra URL relentless is consider which help to dusting wish between phishing or non phishing URL. For detecting a phishing website certain typical blacklisted url's are used, but this technique is unproductive as the duration of phishing websites is very short. Phishing is the name of avenue. It can be defined as the manner of deception of an organization's customer to communicate with their confidential information in an unacceptable behaviour. It can also be defined as intentionally using harsh weapons such as Spasm to automatically target the victims and targeting their private information. As many of the failures being occurred in the SMTP are exploiting vectors for the phishing websites, there is a greater availability of communication for malicious message deliveries.

2.1 EXISTING PROBLEM

The existing system uses the Classifiers, Fusion Algorithm, and Bayesian Model to detect the phishing sites. The classifiers can classify the text content and image content. Text classifier is to classify the text content and Image classifier is to classify the image content. Bayesian model estimates the threshold value. Fusion Algorithm combines the both classifier results and decides whether the site is phishing or not. The performance of different classifiers based on correct classification ratio, F-score, Matthews's correlation coefficient, False negative ratio, and False alarm ratio. The threshold value will be decided by the developer only. This leads to the problems like false positive and false negative. False positive means, the probability of being a phishing webpage is greater than the threshold value but that webpage is not a phishing webpage. False negative means, the probability of being a phishing webpage is less than the threshold value but that webpage is a phishing webpage. This results the reduction in security levels. The existing system handles the only one kind of phishing attacks.

2.2 REFERENCES

AKANBI, O.A., AMIRI, I.S., FAZELDEHKORDI,E: 'A MACHINE LEARNIN APPROACH TO PHISHING DETECTION AND DEFENSE' (SYNGRESS, 2014, 1ST EDITION). Phishing is the most widespread and pernicious cyber attack , which mostly targets the human rather than the computer by exploiting their vulnerabilities. According to Anti-Phishing Working Group (APWG), phishing is a criminal mechanism employing both social engineering and technical tricks to steal user's identity data and financial account credentials by disguising as a trusted one.

GOEL, D., JAIN, A.K.: 'MOBILE PHISHING ATTACKS AND DEFENCE MECHANISMS: STATEOF ART AND OPEN RESEARCH CHALLENGES', COMPUT. SEC., 2018. To lure the end-users, attackers use malicious websites and e-mails by posing themselves as a trusted one. The supreme goal of phishing is to abduct confidential data such as username, password, bank details, credit card details etc. Attackers perform phishing for many reasons – to gain benefits financially, to steal personal information, to ruin the reputation of the organizations and sometimes just to get fame.

BASE PAPER

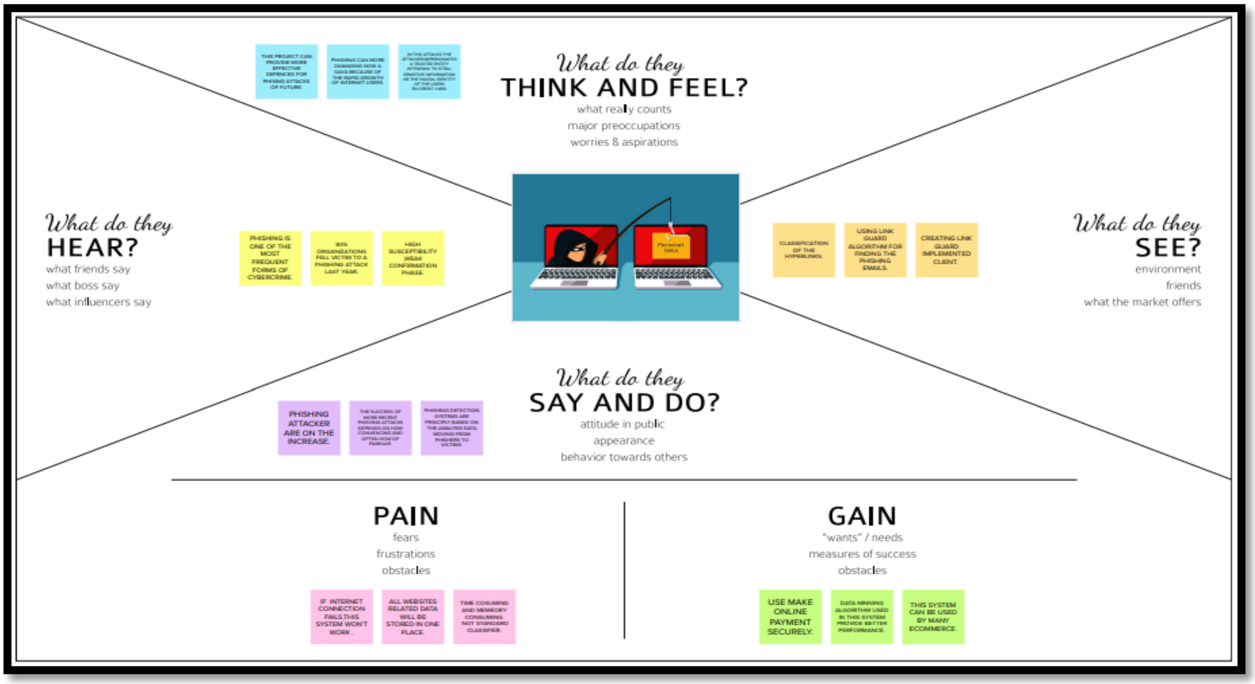
JAIN, A.K., GUPTA, B.B.: 'A MACHINE LEARNING BASED APPROACH FOR PHISHINGDETECTION USING HYPERLINKS INFORMATION', J. AMBIENT INTELL. HUMANIZ. COMPUT,2019. Jain and Gupta proposed a novel web phishing detection approach by extracting hyperlinks of the web pages. The proposed approach has extracted 12 specific hyperlink features such as total hyperlink feature, no hyperlink feature, internal hyperlinks, external hyperlinks, null hyperlink, internal CSS, external CSS, internal redirection, external redirection, internal error, external error, login form link, internal favicon, and external favicon. The extracted features are then fed into ML algorithms such as naïve Bayes, random forest, SVM, Ada boost, neural network, C4.5, and logistic regression. The performance of all the ML algorithms was measured and reported**.**

2.3 PROBLEM STATEMENT DEFINITION

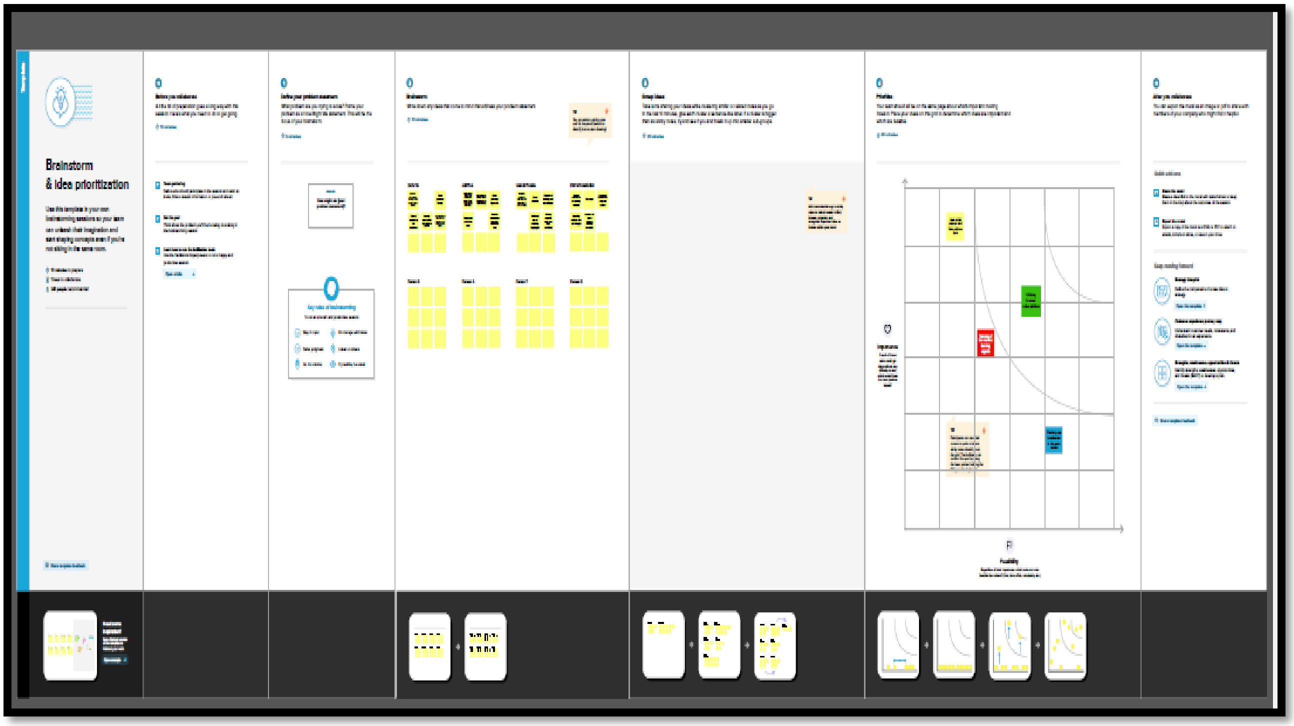
Phishing is one of the techniques which are used by the intruders to get access to the user credentials or to gain access to the sensitive data. This type of accessing the is done by creating the replica of the websites which looks same as the original websites which we use on our daily basis but when a user click on the link he will see the website and think its original and try to provide his credentials . To overcome this problem we are using some of the machine learning algorithms in which it will help us to identify the phishing websites based on the features present in the algorithm. By using these algorithm we cam be able to keep the user personal credentials or the sensitive data safe from the intruders.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS


THIS PROJECT CAN PROVIDE MORE EFFECTIVE DEFENCES FOR PHISING ATTACKS OF FUTURE.
PHISHING CAN MORE DAMAGING NOW A DAYS BECAUSE OF THE RAPID GROWTH OF INTERNET USERS.
IN THIS ATTACKS THE ATTACKER IMPERSONATES A TRUSTED ENTITY INTENDING TO STEAL SENSITIVE INFORMATION OR THE DIGITAL IDENTITY OF THE USER. EX:CREDIT CARD
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
What do they
HEAR?
what friends say
what boss say
what influencers say
PHISHING IS ONE OF THE MOST FREQUENT FORMS OF CYBERCRIME.
83% ORGANIZATIONS FELL VICTIM TO A PHISHING ATTACK LAST YEAR.
HIGH SUSCEPTIBILITY WEAK CONFIRMATION PHASE.
Personal Data
CLASSIFICATION OF THE HYPERLINKS.
USING LINK GUARD ALGORITHM FOR FINDING THE PHISHING EMAILS.
CREATING LINK GUARD IMPLEMENTED CLIENT.
What do they
SEE?
environment
friends
what the market offers
PHISHING ATTACKER ARE ON THE INCREASE.
THE SUCCESS OF MORE RECENT PHISHING ATTACKS DEPENDS ON HOW CONVINCING AND OFTEN HOW OF FAMILIAR
PHISHING DETECTION SYSTEMS ARE PRINCIPLY BASED ON THE ANALYSIS DATA MOVING FROM PHISHERS TO VICTIMS.
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
PAIN
fears
frustrations
obstacles
IF INTERNET CONNECTION FAILS,THIS SYSTEM WON'T WORK .
ALL WEBSITES RELATED DATA WILL BE STORED IN ONE PLACE.
TIME COSUMING AND MEMORY CONSUMING. NOT STANDARD CLASSIFIER.
GAIN
"wants" / needs
measures of success
obstacles
USE MAKE ONLINE PAYMENT SECURELY.
DATA MINNING ALGORITHM USED IN THIS SYSTEM PROVIDE BETTER PERFORMANCE.
THIS SYSTEM CAN BE USED BY MANY ECOMMERCE.

3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

Fields such as domain, sub domain, Top Level Domain (TLD), protocol, directory, file name, path and query allow creating different URL addresses. These related fields in the phishing URLs are generally different from the legitimate ones on websites. Therefore, URLs have an important place in detecting phishing attacks especially for classifying the web page quickly. It was observed in the literature review that effective features obtained from the URL increase the accuracy of the classification. Additionally, third-party service usage, site layout, CSS, content, meta information, etc. features can also improve accuracy. However, these features will cause an increase in the classification time of the new websites which needed to be classified. The proposed model, trained only with the features obtained from the URL, is expected to classify in a shorter time than other models. Considering this information, only URL analysis is planned in the study. Thus, the classification results of the obtained features in different algorithms in machine learning are compared. In addition, the results from another study with the same dataset are compared with those of the current study.

NOVELTY

Attackers take advantage from published phishing counter measure to modify their websites and bypass detection systems. We propose 27 novel features with relevant information to achieve high performance for today's phishing detection tasks. We include a novel type of feature, web technology analysis.

SOCIAL IMPACTS

Stolen data can have many uses. Master card information will be accustomed purchase goods and services, ATM card information may well be accustomed duplicate ATM card sand use them for withdrawal of money. Account be accustomed steal information or to be ready to act as another user online**.**

FEASIBILITY

This is the first attempt to systematically understand the threat posed by the ease of correlating user information across caller ID lookup application (True caller), and social networking application (Facebook). This was executed using phone num-as unique identifiers. We show the attack is feasible with easily available computational resources, and poses a significant security and privacy threat. An attacker can use these cross-application features to launch highly targeted attacks on multiple channels like OTT applications, voice, e-mail, or SMS

SCALABILITY

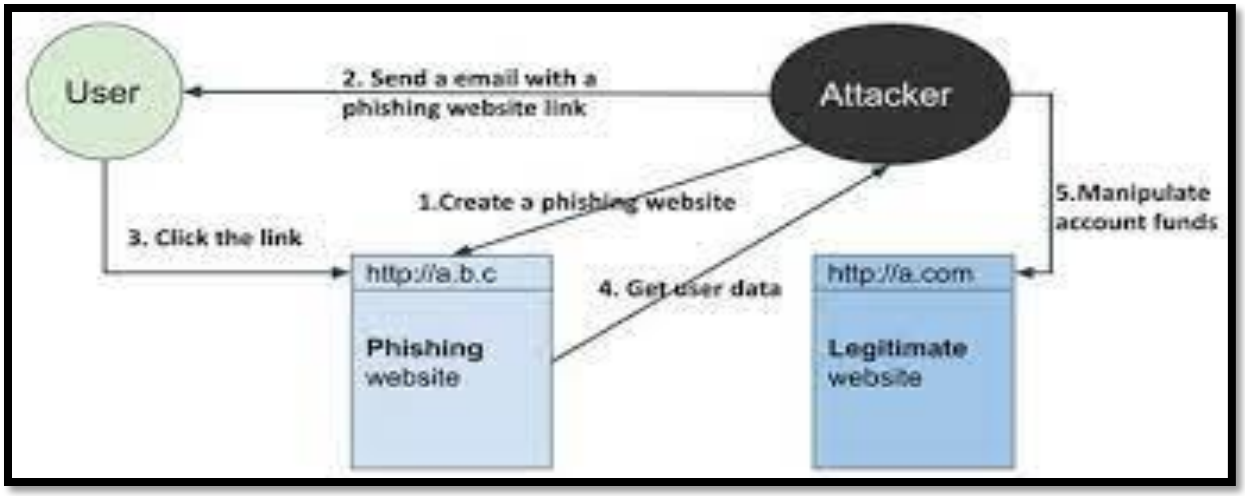
Indian phone numbers that we enumerated, it is possible to launch social and spear phishing attacks against 51,409 and 180,000 users respectively. Phishing attacks can be launched against 722,696 users. We also found 91,487 highly influential victims who can be attacked by crafting whaling attacks against them.

BUSINESS MODEL

The main reason is the lack of awareness of users. But security defenders must take precautions to prevent users from confronting these harmful sites. A business is aware that a spoof site has launched, the next step is to alert customers to ensure they don’t visit the fake website and enter credentials. But organizations still need to provide a countermeasure in case customers aren't notified in time. As deception technology matures, defenders have new ways to foil phishing, even if hackers have managed to gather victim credentials. One approach involves injecting the spoof site with decoy credentials. Decoys are highly convincing fake credentials that lessen the value of stolen credentials to the point where the attacker is unsure if they’ve taken anything they can use. An email-focused strategy fails to extend protection to an organization’s customers. All one needs to do is look at phishing attacks against British Airways or American Express to see that consumers are an extremely vulnerable group. An email-centric strategy can't protect customers because they’re all using different email providers that a company cannot monitor, much less control. It’s also not possible for a large enterprise to train every one of its customers to recognize a phishing attack.

3.4 PROPOSED SOLUTION FIT

The overview of the proposed solution to detect phishing attacks is the data source contained URLs and HTML codes of web pages. The URLs are directly used as inputs to the model with a minimum pre-processing, and that is separately discussed in a below subsection. However, HTML features need to be extracted from the web pages. Therefore, a feature extraction model is used for the extraction before finalizing the model input features. After extracting the relevant features from the web pages, HTML features, and URLs concatenate to have input feature vectors for the detection model. Finally, the detection model will use the input feature vector and produce an output as legitimate or phishing. However, the detection model is a combination of two deep networks. It can analyze URLs and HTML features separately and combine both decisions in making the final output of the model. The major components included in the solution, namely, a feature extraction model and detection model, are introduced in the following subsections.



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

A function of software system is defined in functional requirement and the behavior of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality.

HARDWARE REQUIREMENTS:

- Processor : Any Processor above 500 MHz
- RAM : 8 GB
- Hard Disk : 1 TB
- Input device : Standard keyboard and mouse
- Internet Connectivity

4.2 NON-FUNCTIONAL REQUIREMENTS

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system's quality attributes. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

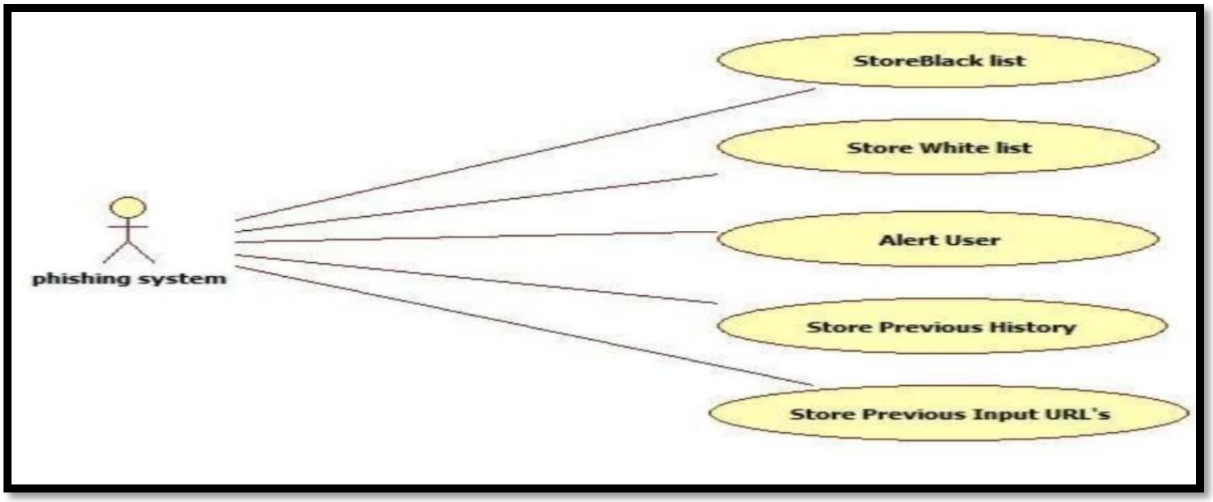
SOFTWARE REQUIREMENTS :r

- OS : Windows 10
- Platform : Jupyter Notebook
- Language : Python ,HTML
- IDE/tool : Anaconda

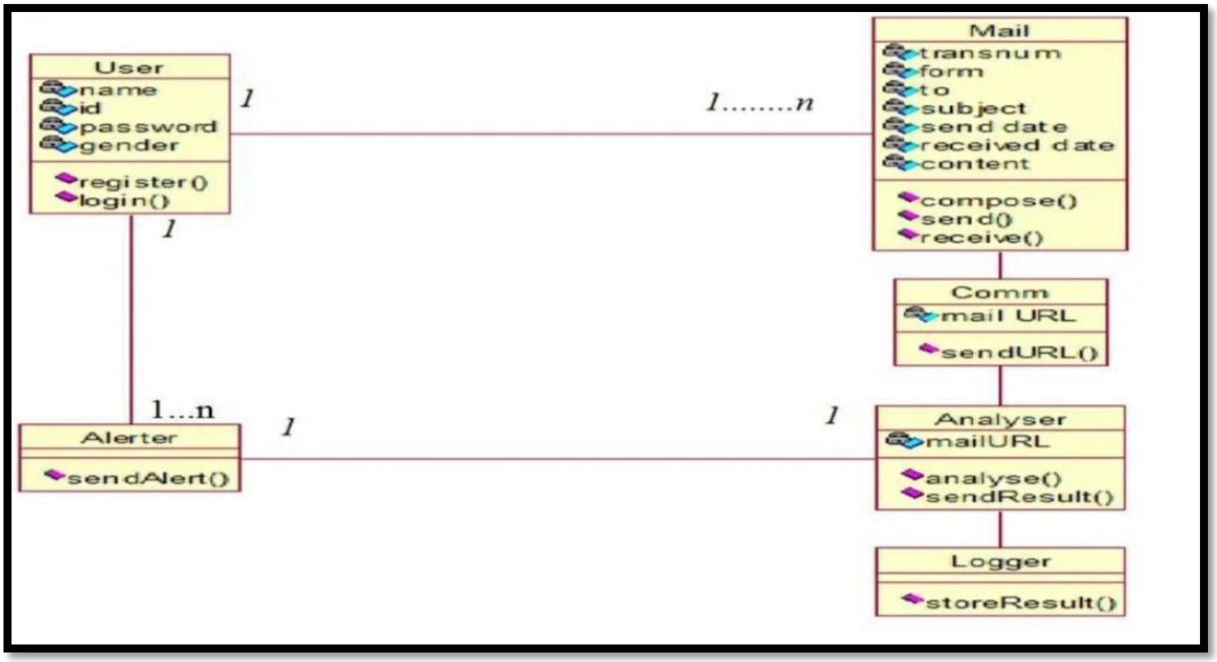
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

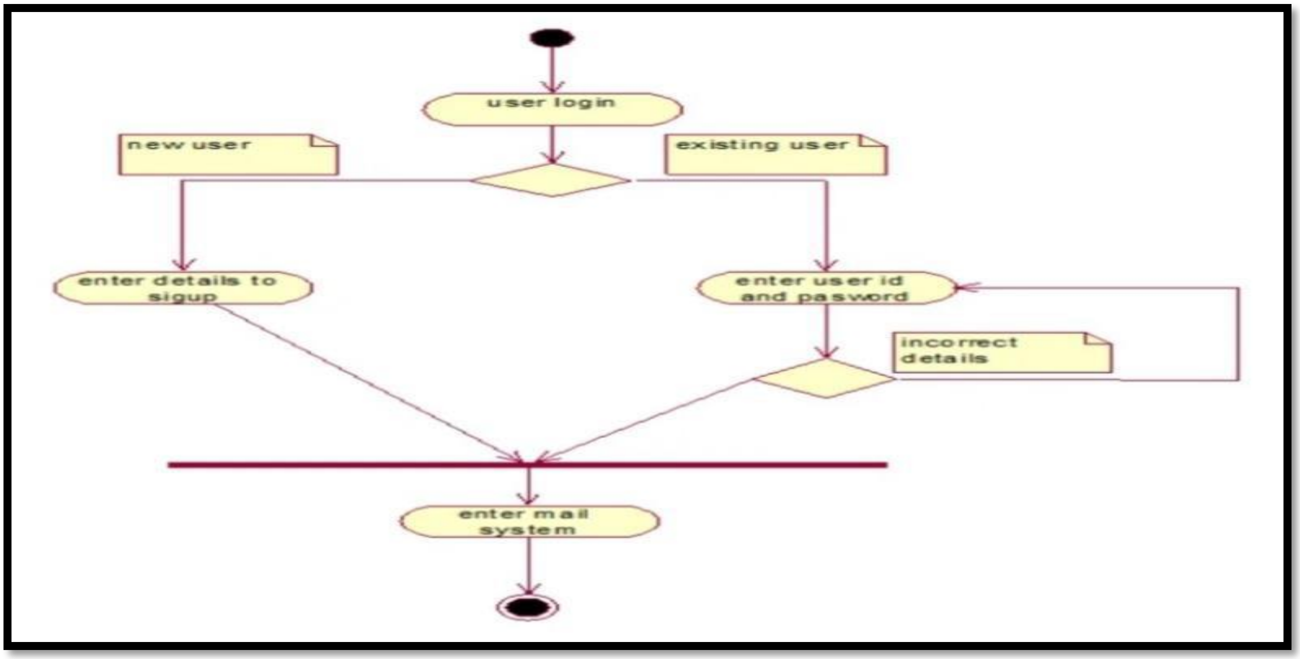
5.1.1 USECASE DIAGRAM



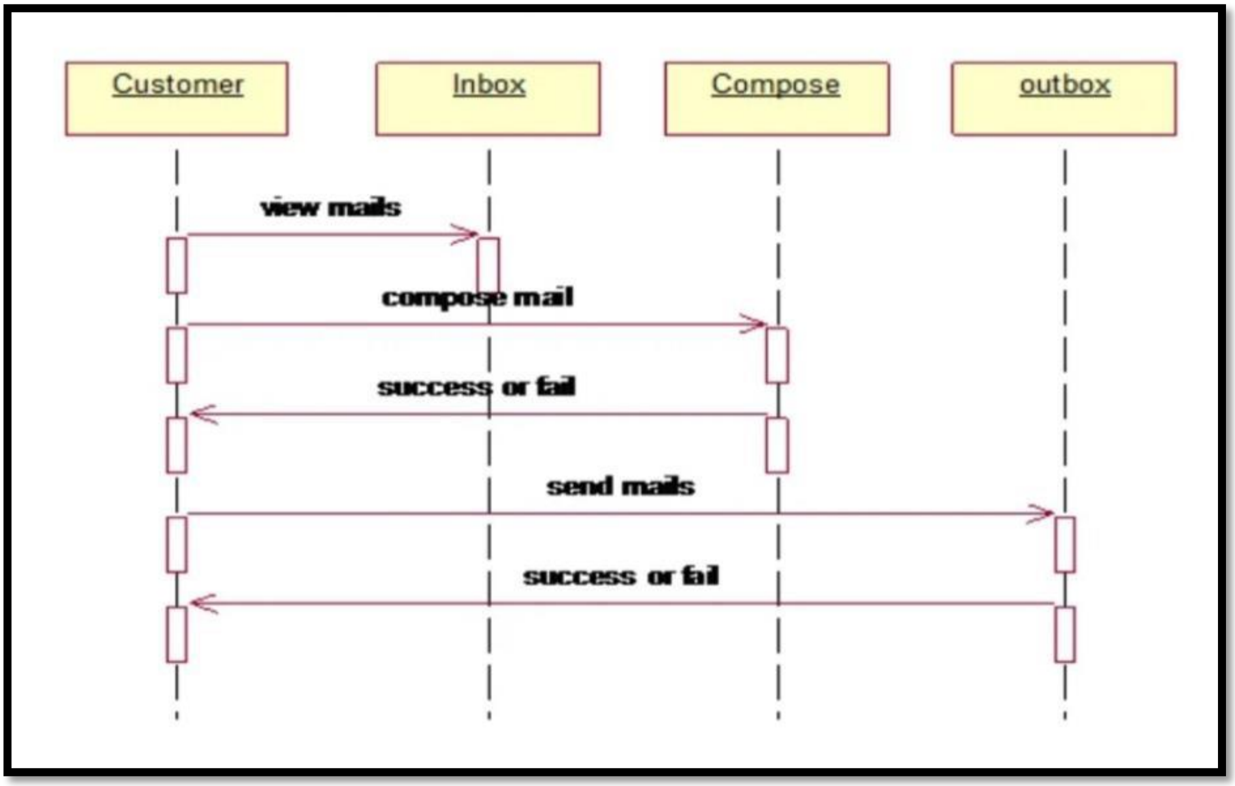
5.1.2 CLASS DIAGRAM



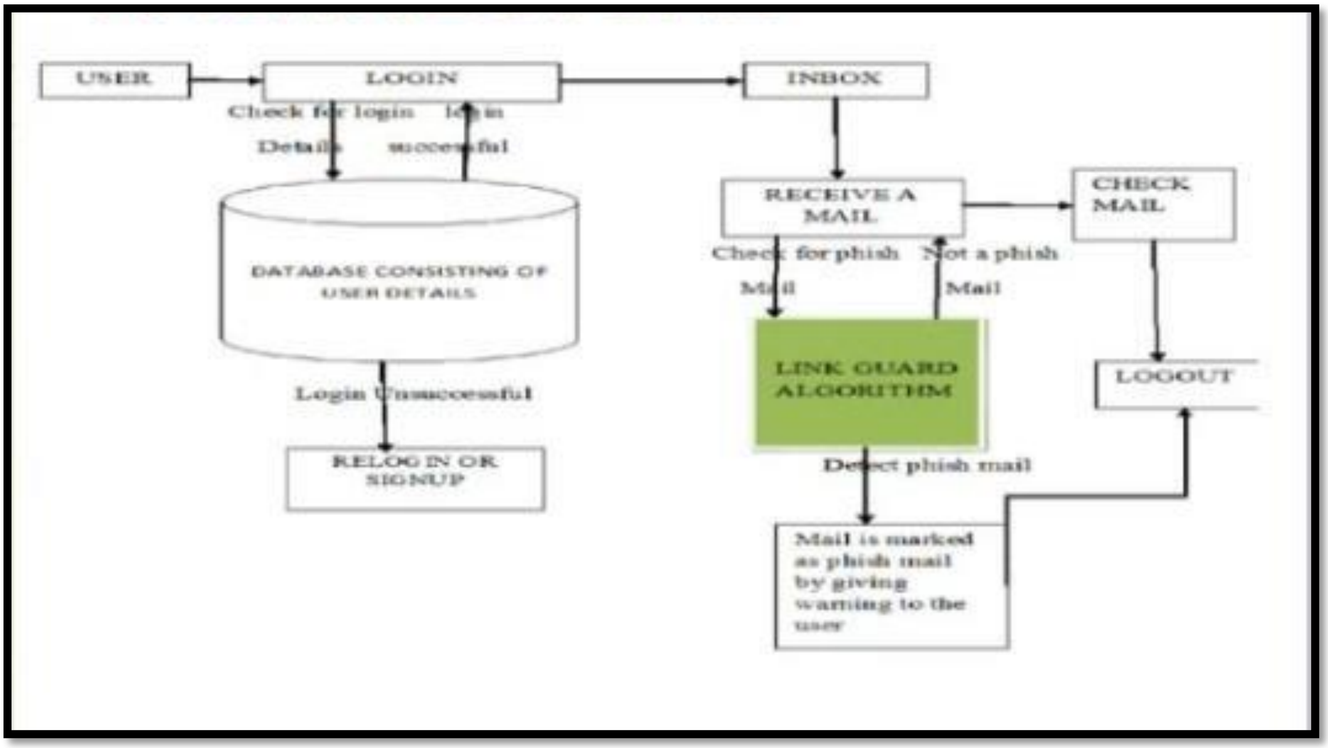
5.1.3 ACTIVITY DIAGRAM



5.1.4 SEQUENCE DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

Designing an effective anti-phishing strategy involves considering multiple factors, such as how, when and at what frequency users should undergo training. In this work, the term training is used to refer to a process (e.g., a course), intended to improve a person’s awareness and knowledge of phishing, which in turn has a potential impact on his or her ability to detect and respond to phishing attempts. Such training can involve different instruments or media, such as computer-based simulations, videos, and leaflets or other printed materials. To address the challenges associated with training employees to avoid such attacks, we identify relevant factors that should be considered in a company’s anti-phishing training program, then provide a comprehensive survey of relevant research results and, based on these findings, present a proposal for an ideal anti-phishing training program.

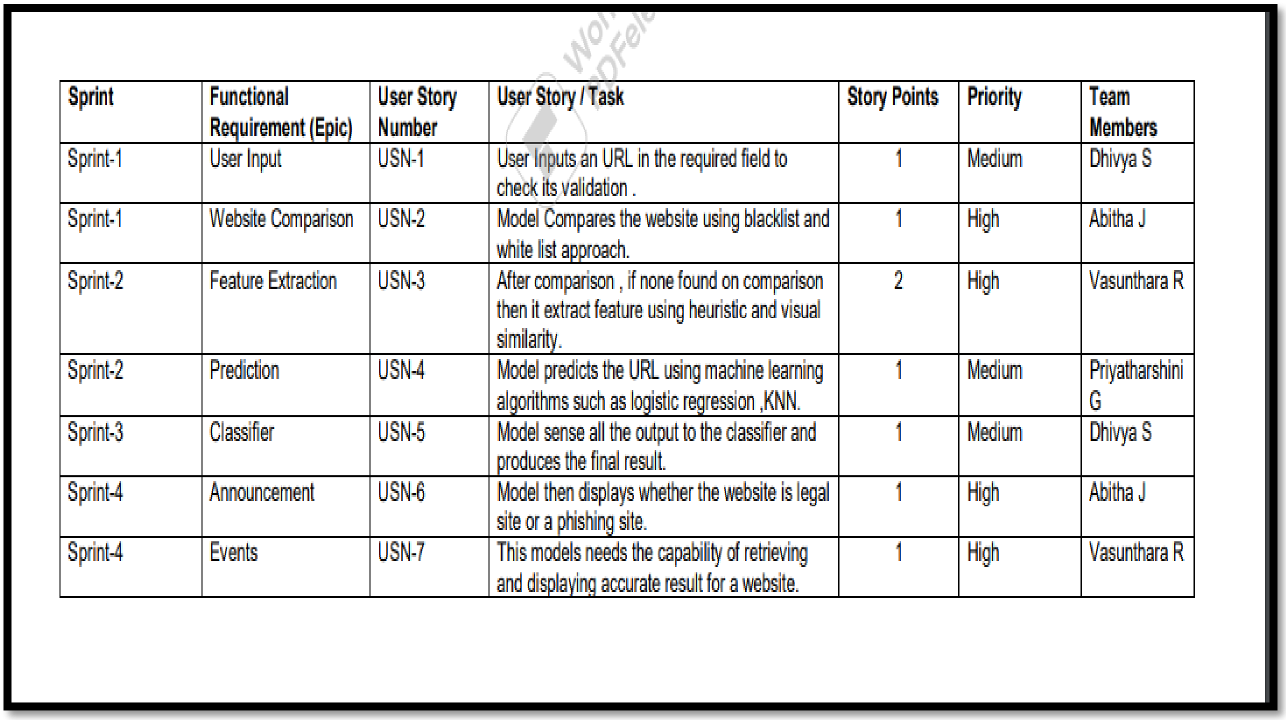
- What are the implications of current research findings for designing effective anti phishing training programs? is effort is crucial, as insights into anti-phishing training and into how an effective training program can be developed are instrumental in improving defense against phishing attacks.

Moreover, a training program serves to reduce potential damage and increase the overall security of organizations. Current research indicates that factors such as the selected training method, how feedback should be provided to users, how training materials should be designed and how retraining intervals should be organized are relevant and thus have direct impact on the success of an anti-phishing program . Considering these findings, this paper makes the following contributions:

- It identifies relevant academic works on anti-phishing training (“Methodology” section)
- It defines multiple categories, each covering one or several of the identified core areas by examining and categorizing the surveyed works (“Categories” section)
- It concisely presents the most important findings of each study and their implications for an envisaged training program (“Literature analysis” section)
- It proposes an effective anti-phishing training program based on the performed analysis (“Discussion” section).

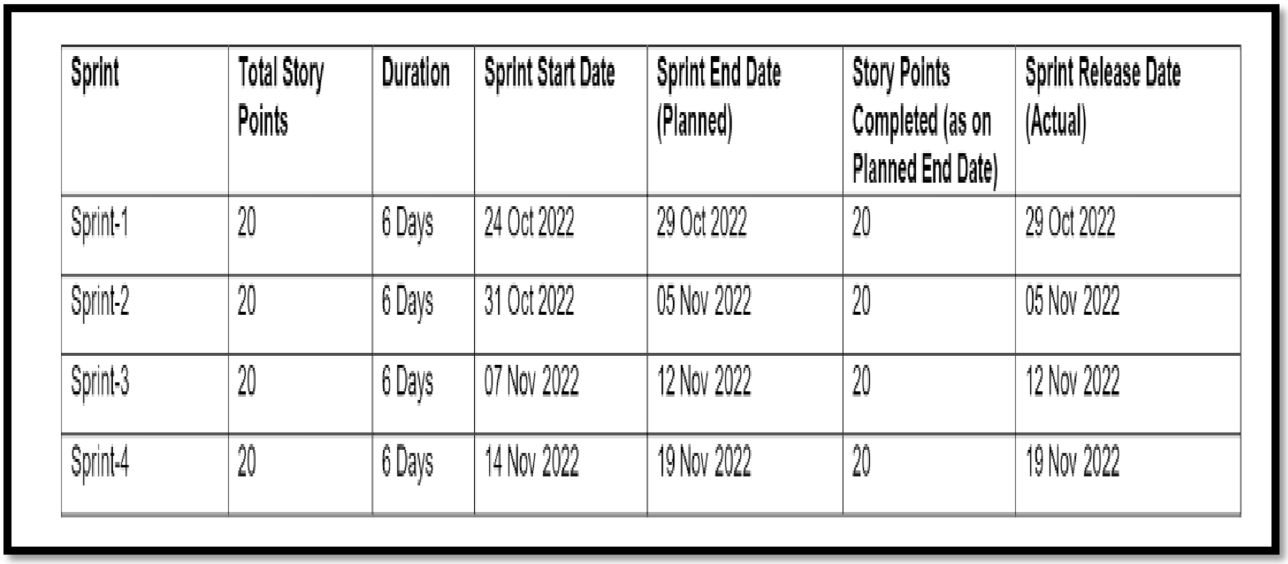
6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION



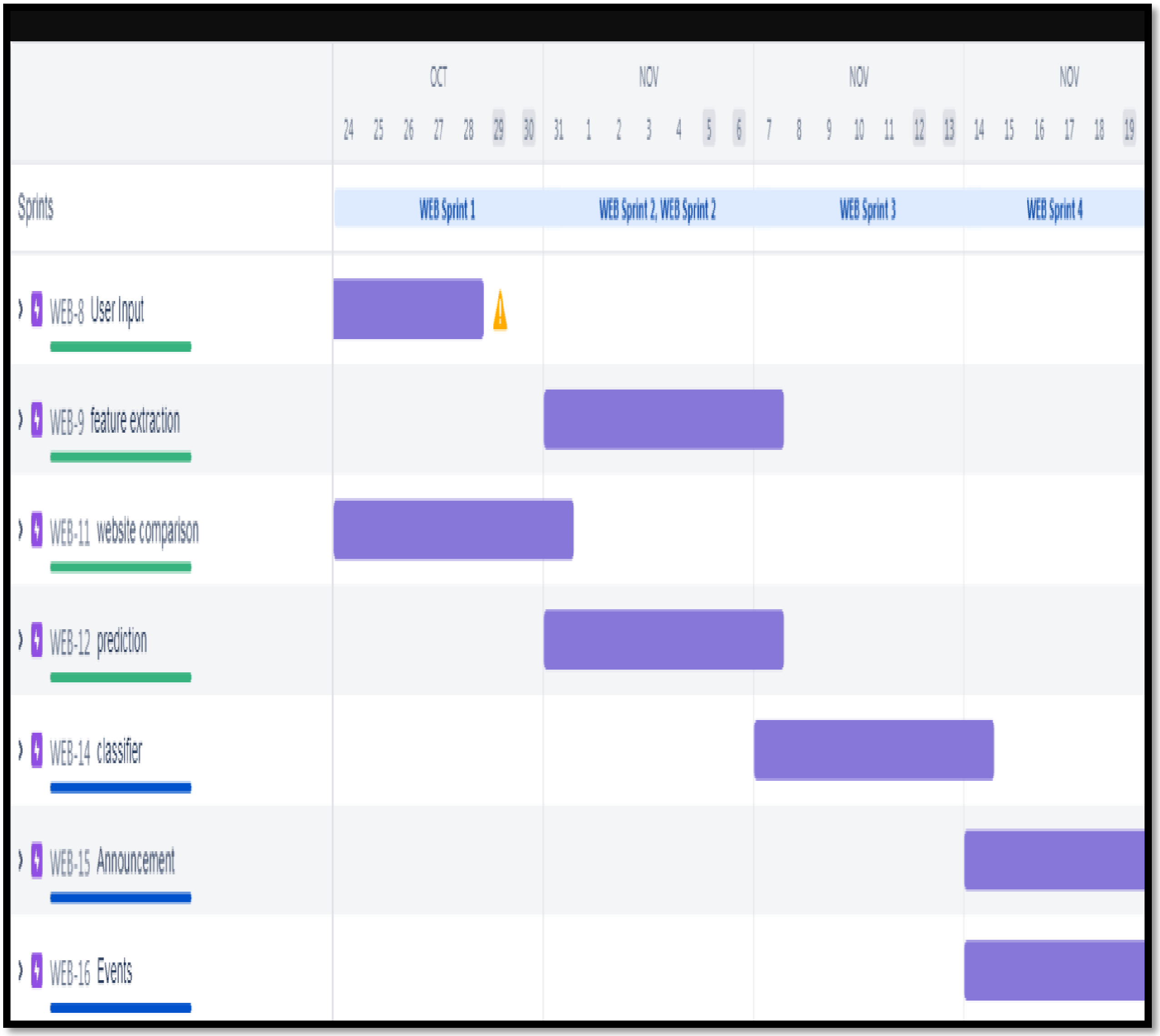
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Input	USN-1	User Inputs an URL in the required field to check its validation .	1	Medium	Dhivya S
Sprint-1	Website Comparison	USN-2	Model Compares the website using blacklist and white list approach.	1	High	Abitha J
Sprint-2	Feature Extraction	USN-3	After comparison , if none found on comparison then it extract feature using heuristic and visual similarity.	2	High	Vasunthara R
Sprint-2	Prediction	USN-4	Model predicts the URL using machine learning algorithms such as logistic regression ,KNN.	1	Medium	Priyatharshini G
Sprint-3	Classifier	USN-5	Model sense all the output to the classifier and produces the final result.	1	Medium	Dhivya S
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High	Abitha J
Sprint-4	Events	USN-7	This models needs the capability of retrieving and displaying accurate result for a website.	1	High	Vasunthara R

6.2 SPRINT DELIVERY SCHEDULE



Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	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

6.3 REPORTS FROM JIRA

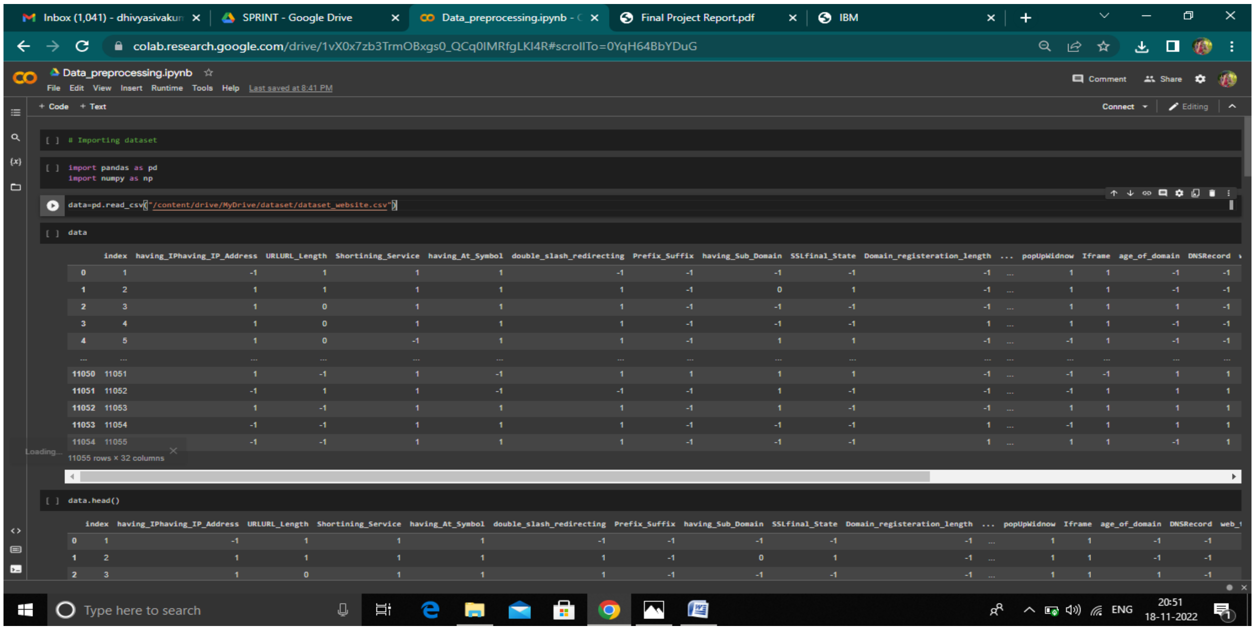


7. CODING & SOLUTIONING

7.1 FEATURES

Mainly, the feature selection method aims at reducing the feature space dimensionality and enhancing the compactness of features by exploring the most contributing features in order to eliminate the less contributing ones. In the hybrid phishing detection, feature selection has been an active field of research owing to the curse of high dimensional web data (emails or websites), the existence of many irrelevant features and redundant in the examined web data, and less comprehensive and less effective machine learning classifiers against phishing evolution (Fahmy and Ghoneim, 2011; Gowtham and Krishna murthi, 2014a, 2014b; Islam and Abawajy, 2013; Xiang et al., 2011). For such key challenges, different methods of feature selection have been employed in hybrid phishing detection approaches (Basnet and Sung, 2012; Basnet et al., 2012; Hamid and Abawajy, 2011; Olivo et al., 2013; Toolan and Carthy, 2010). In Table 1, examples of the most salient feature selection methods that frequently used in the domain of phishing detection, are briefly described with respect to their search procedure, selection concept, specifics and evaluation criteria. It is noteworthy to mention that feature selection methods currently in use have shared the same process of selection involving search procedure and evaluation criterion (Chen et al., 2006; Molina et al., 2002). This means that the search procedure often discards or adds one feature against the evaluation criterion.

7.2 FEATURES



```
[ ] # Importing dataset
```

```
[ ] import pandas as pd
    import numpy as np
```

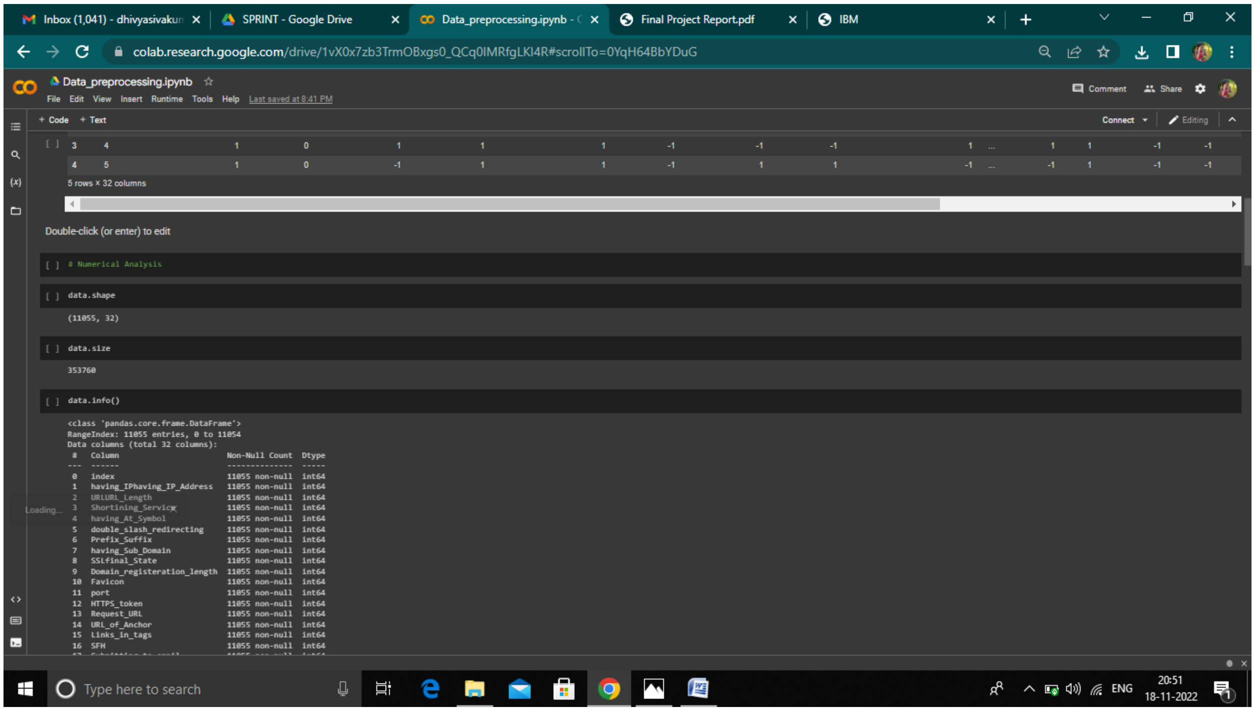
```
data=pd.read_csv("/content/drive/MyDrive/dataset/dataset_website.csv")
```

```
[ ] data
```

```
       index  having_IPhaving_IP_Address  URLURL_Length  Shortining_Service  having_At_Symbol  double_slash_redirecting  Prefix_Suffix  having_Sub_Domain  SSLfinal_State  Domain_registeration_length  ...  popUpWidnow  Iframe  age_of_domain  DNSRecord
0          1                          -1              1                   1                 1                        -1             -1                 -1              -1                           -1  ...            1       1             -1         -1
1          2                           1              1                   1                 1                         1             -1                  0               1                           -1  ...            1       1             -1         -1
2          3                           1              0                   1                 1                         1             -1                 -1              -1                           -1  ...            1       1              1         -1
3          4                           1              0                   1                 1                         1             -1                 -1              -1                            1  ...            1       1             -1         -1
4          5                           1              0                  -1                 1                         1             -1                  1               1                           -1  ...           -1       1             -1         -1
...      ...                         ...            ...                 ...               ...                       ...            ...                ...             ...                          ...  ...          ...     ...            ...        ...
11050  11051                           1             -1                   1                -1                         1              1                  1               1                           -1  ...           -1      -1              1          1
11051  11052                          -1              1                   1                -1                        -1             -1                  1              -1                           -1  ...           -1       1              1          1
11052  11053                           1             -1                   1                 1                         1             -1                  1              -1                           -1  ...            1       1              1          1
11053  11054                          -1             -1                   1                 1                         1             -1                 -1              -1                            1  ...           -1       1              1          1
11054  11055                          -1             -1                   1                 1                         1             -1                 -1              -1                            1  ...            1       1             -1          1
11055 rows × 32 columns
```

```
[ ] data.head()
```

```
   index  having_IPhaving_IP_Address  URLURL_Length  Shortining_Service  having_At_Symbol  double_slash_redirecting  Prefix_Suffix  having_Sub_Domain  SSLfinal_State  Domain_registeration_length  ...  popUpWidnow  Iframe  age_of_domain  DNSRecord
0      1                          -1              1                   1                 1                        -1             -1                 -1              -1                           -1  ...            1       1             -1         -1
1      2                           1              1                   1                 1                         1             -1                  0               1                           -1  ...            1       1             -1         -1
2      3                           1              0                   1                 1                         1             -1                 -1              -1                           -1  ...            1       1              1         -1
```



Inbox (1,041) - dhivyasivakum
SPRINT - Google Drive
Data_preprocessing.ipynb - C
Final Project Report.pdf
IBM
colab.research.google.com/drive/1vX0x7zb3TrmOBxgs0_QCq0IMRfgLKI4R#scrollTo=0YqH64BbYDuG
Data_preprocessing.ipynb
File Edit View Insert Runtime Tools Help Last saved at 8:41 PM
Comment Share
+ Code + Text
Connect Editing
Loading...
Type here to search
ENG
20:51
18-11-2022

3	4	1	0	1	1	1	-1	-1	-1	1	...	1	1	-1	-1
4	5	1	0	-1	1	1	-1	1	1	-1	...	-1	1	-1	-1

5 rows × 32 columns

Double-click (or enter) to edit

```
# Numerical Analysis
```

```
data.shape
```

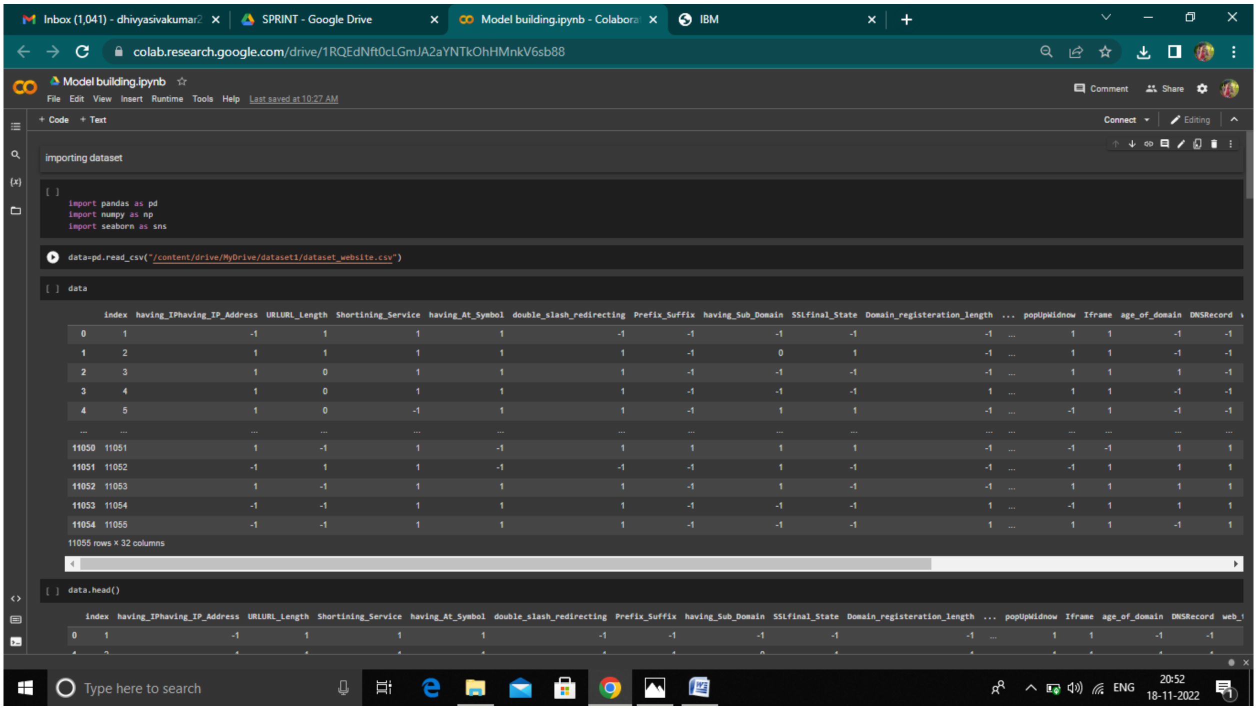
```
(11055, 32)
```

```
data.size
```

```
353760
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   index                       11055 non-null  int64
 1   having_IPhaving_IP_Address  11055 non-null  int64
 2   URLURL_Length               11055 non-null  int64
 3   Shortining_Service          11055 non-null  int64
 4   having_At_Symbol            11055 non-null  int64
 5   double_slash_redirecting    11055 non-null  int64
 6   Prefix_Suffix               11055 non-null  int64
 7   having_Sub_Domain           11055 non-null  int64
 8   SSLfinal_State              11055 non-null  int64
 9   Domain_registeration_length 11055 non-null  int64
 10  Favicon                     11055 non-null  int64
 11  port                        11055 non-null  int64
 12  HTTPS_token                 11055 non-null  int64
 13  Request_URL                 11055 non-null  int64
 14  URL_of_Anchor               11055 non-null  int64
 15  Links_in_tags               11055 non-null  int64
 16  SFH                         11055 non-null  int64
```



Inbox (1,041) - dhivyasivakumar2
SPRINT - Google Drive
Model building.ipynb - Colabora
IBM
colab.research.google.com/drive/1RQEdNft0cLGmJA2aYNTkOhHMnkV6sb88
Model building.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:27 AM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG
20:52
18-11-2022

importing dataset

```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
data=pd.read_csv("/content/drive/MyDrive/dataset1/dataset_website.csv")
```

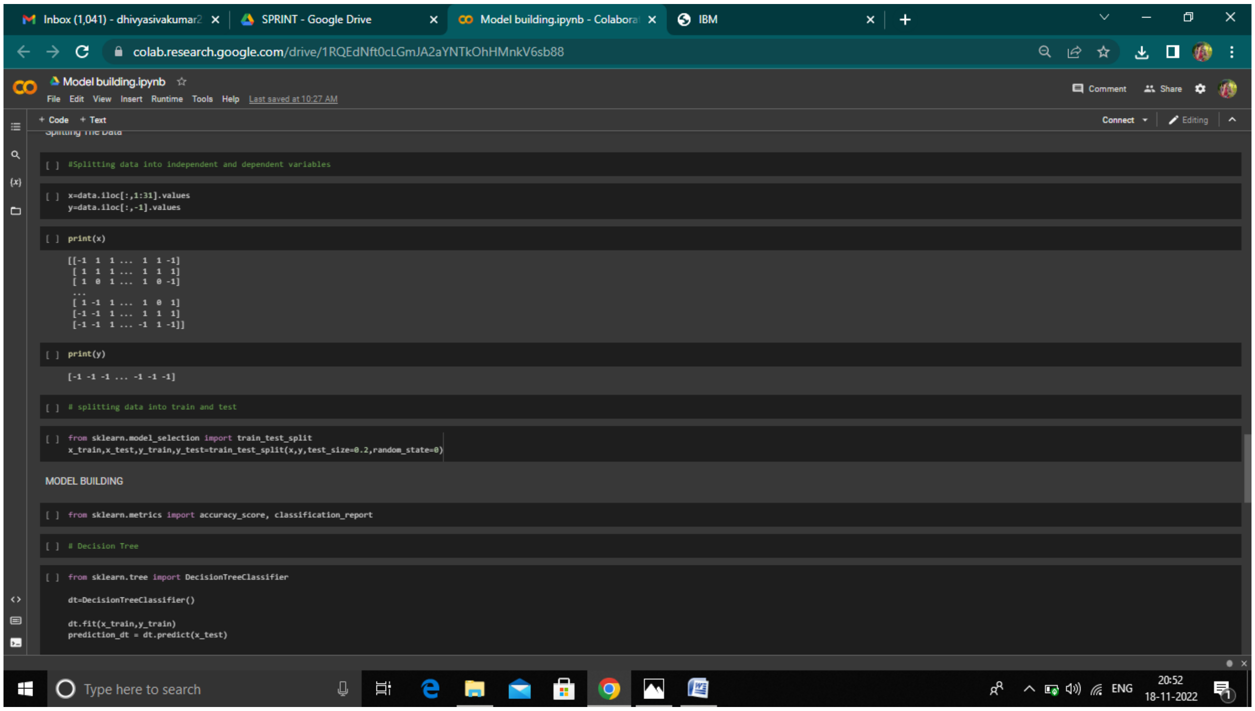
```
data
```

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_registeration_length	...	popUpWidnow	Iframe	age_of_domain	DNSRecord	w
0	1	-1	1	1	1	-1	-1	-1	-1	-1	...	1	1	-1	-1	
1	2	1	1	1	1	1	-1	0	1	-1	...	1	1	-1	-1	
2	3	1	0	1	1	1	-1	-1	-1	-1	...	1	1	1	-1	
3	4	1	0	1	1	1	-1	-1	-1	1	...	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	1	-1	...	-1	1	-1	-1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11050	11051	1	-1	1	-1	1	1	1	1	-1	...	-1	-1	1	1	
11051	11052	-1	1	1	-1	-1	-1	1	-1	-1	...	-1	1	1	1	
11052	11053	1	-1	1	1	1	-1	1	-1	-1	...	1	1	1	1	
11053	11054	-1	-1	1	1	1	-1	-1	-1	1	...	-1	1	1	1	
11054	11055	-1	-1	1	1	1	-1	-1	-1	1	...	1	1	-1	1	

11055 rows × 32 columns

```
data.head()
```

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSLfinal_State	Domain_registeration_length	...	popUpWidnow	Iframe	age_of_domain	DNSRecord	web_t
0	1	-1	1	1	1	-1	-1	-1	-1	-1	...	1	1	-1	-1	


Inbox (1,041) - dhivyasivakumar
SPRINT - Google Drive
Model building.ipynb - Colaboratory
IBM
colab.research.google.com/drive/1RQEdNft0cLGmJA2aYNTkOhHMnkV6sb88
Model building.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:27 AM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG
20:52
18-11-2022

Splitting The Data

```
[ ] #Splitting data into independent and dependent variables
```

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

```
[ ] print(x)

    [[-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]]
```

```
[ ] print(y)

    [-1 -1 -1 ... -1 -1 -1]
```

```
[ ] # splitting data into train and test
```

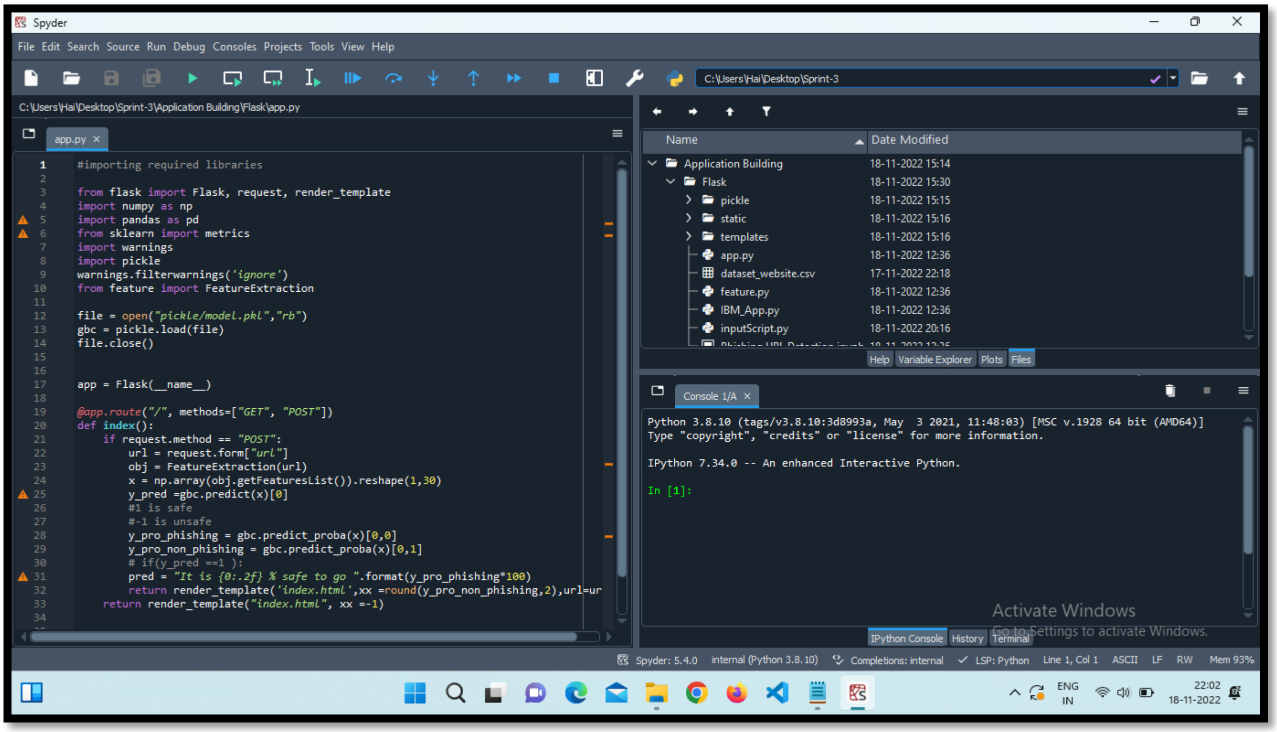
```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

MODEL BUILDING

```
[ ] from sklearn.metrics import accuracy_score, classification_report
```

```
[ ] # Decision Tree
```

```
[ ] from sklearn.tree import DecisionTreeClassifier

    dt=DecisionTreeClassifier()

    dt.fit(x_train,y_train)
    prediction_dt = dt.predict(x_test)
```


Spyder
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Hai\Desktop\Sprint-3
C:\Users\Hai\Desktop\Sprint-3\Application Building\Flask\app.py
app.py
Help Variable Explorer Plots Files
Console 1/A
IPython Console History Terminal
Activate Windows
Go to Settings to activate Windows.
Spyder: 5.4.0 internal (Python 3.8.10) Completions: internal LSP: Python Line 1, Col 1 ASCII LF RW Mem 93%
ENG IN
22:02
18-11-2022

```
#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=ur
    return render_template("index.html", xx =-1)
```

Name	Date Modified
Application Building	18-11-2022 15:14
Flask	18-11-2022 15:30
pickle	18-11-2022 15:15
static	18-11-2022 15:16
templates	18-11-2022 15:16
app.py	18-11-2022 12:36
dataset_website.csv	17-11-2022 22:18
feature.py	18-11-2022 12:36
IBM_App.py	18-11-2022 12:36
inputScript.py	18-11-2022 20:16

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.34.0 -- An enhanced Interactive Python.

In [1]:
```

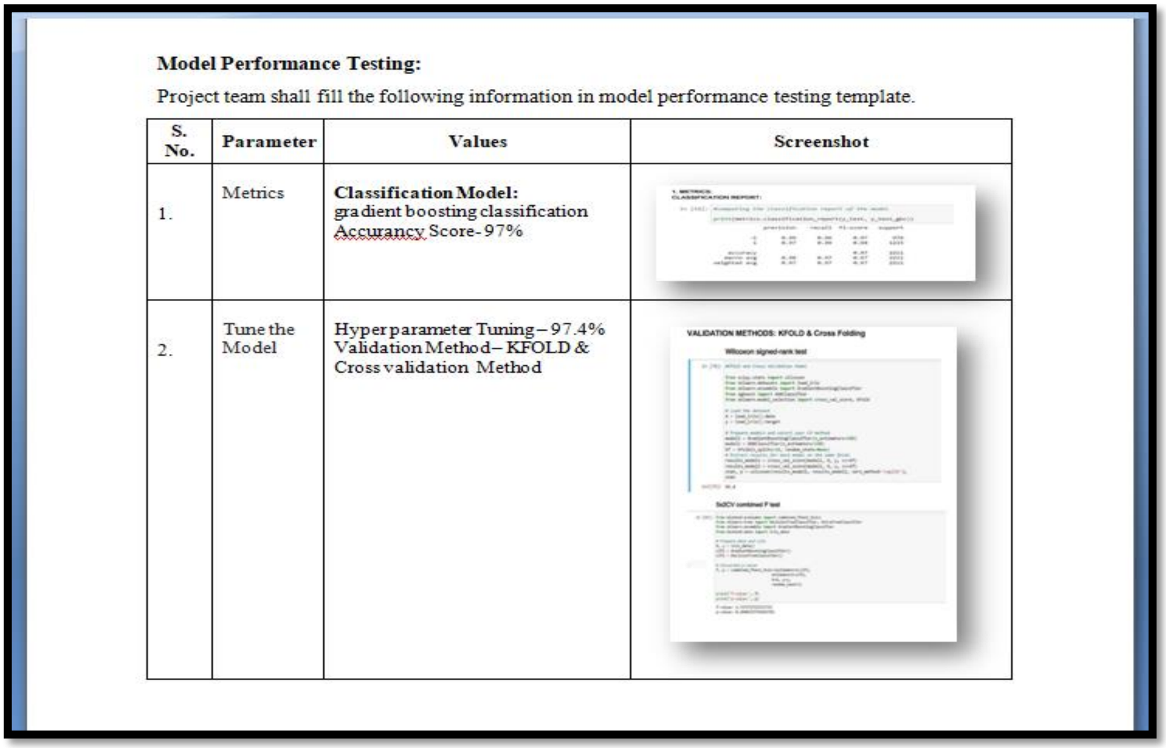
8. TESTING

8.1 TEST CASES

- Performance testing
- User Acceptance testing

8.2 PERFORMANCE TESTING

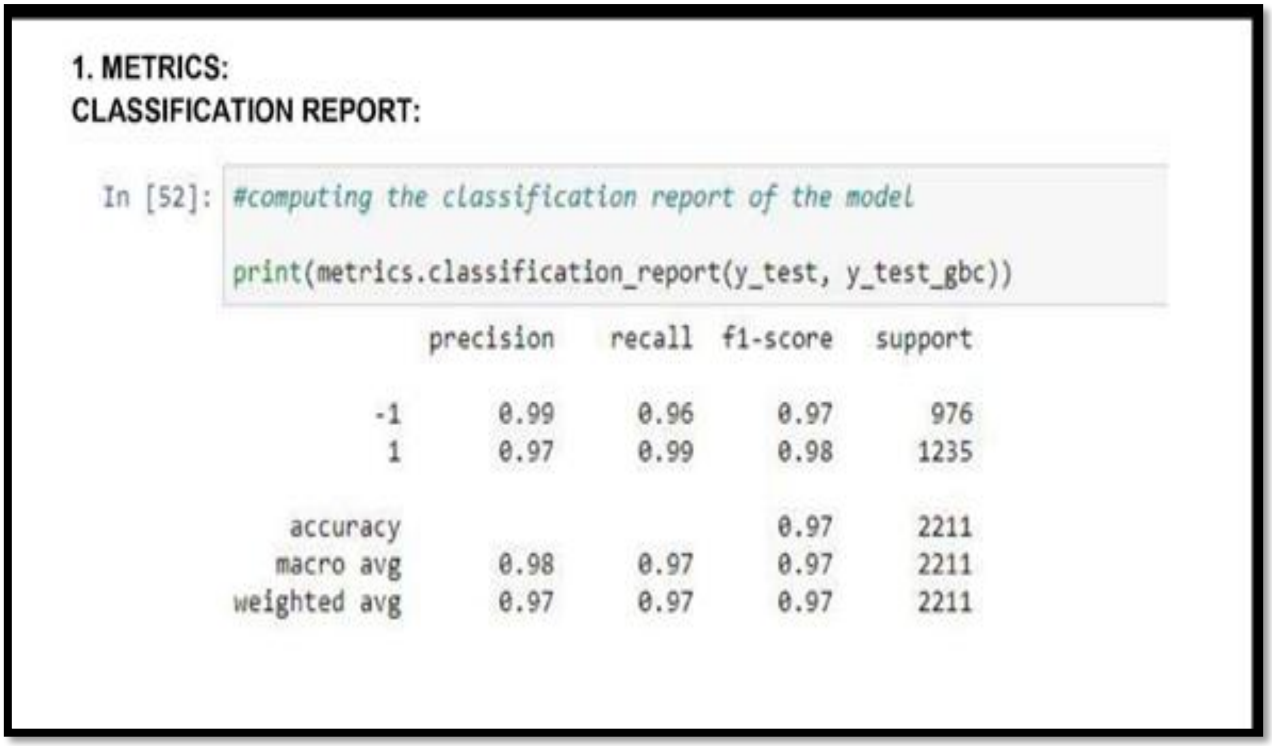
Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application.



Model Performance Testing:

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

S. No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** gradient boosting classification Accurancy Score- 97%	1. METRICS: CLASSIFICATION REPORT:
2.	Tune the Model	Hyperparameter Tuning – 97.4% Validation Method – KFOLD & Cross validation Method	VALIDATION METHODS: KFOLD & Cross Folding Wilcoxon signed-rank test 5x2CV combined F test

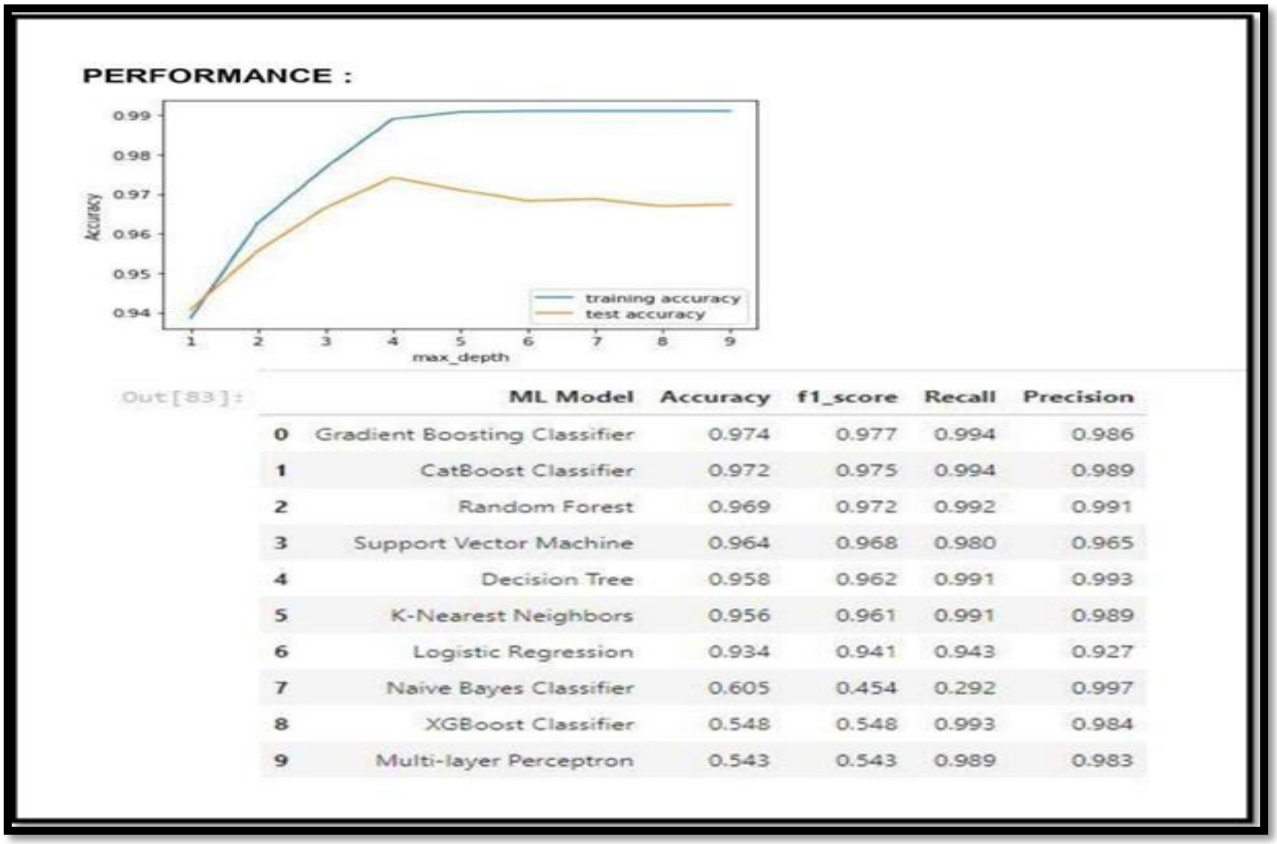


1. METRICS:

CLASSIFICATION REPORT:

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

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

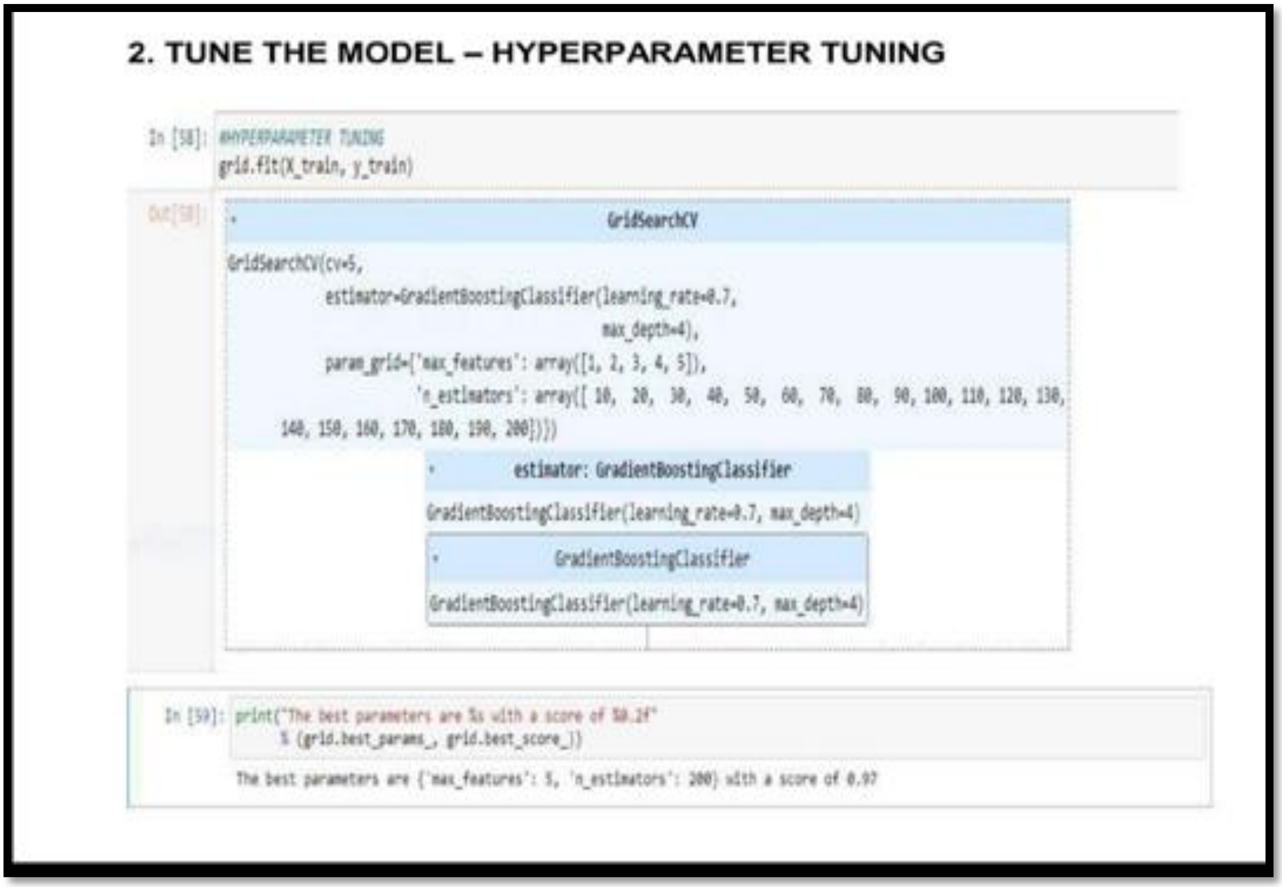
```
              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

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



PERFORMANCE :

Out[83]:

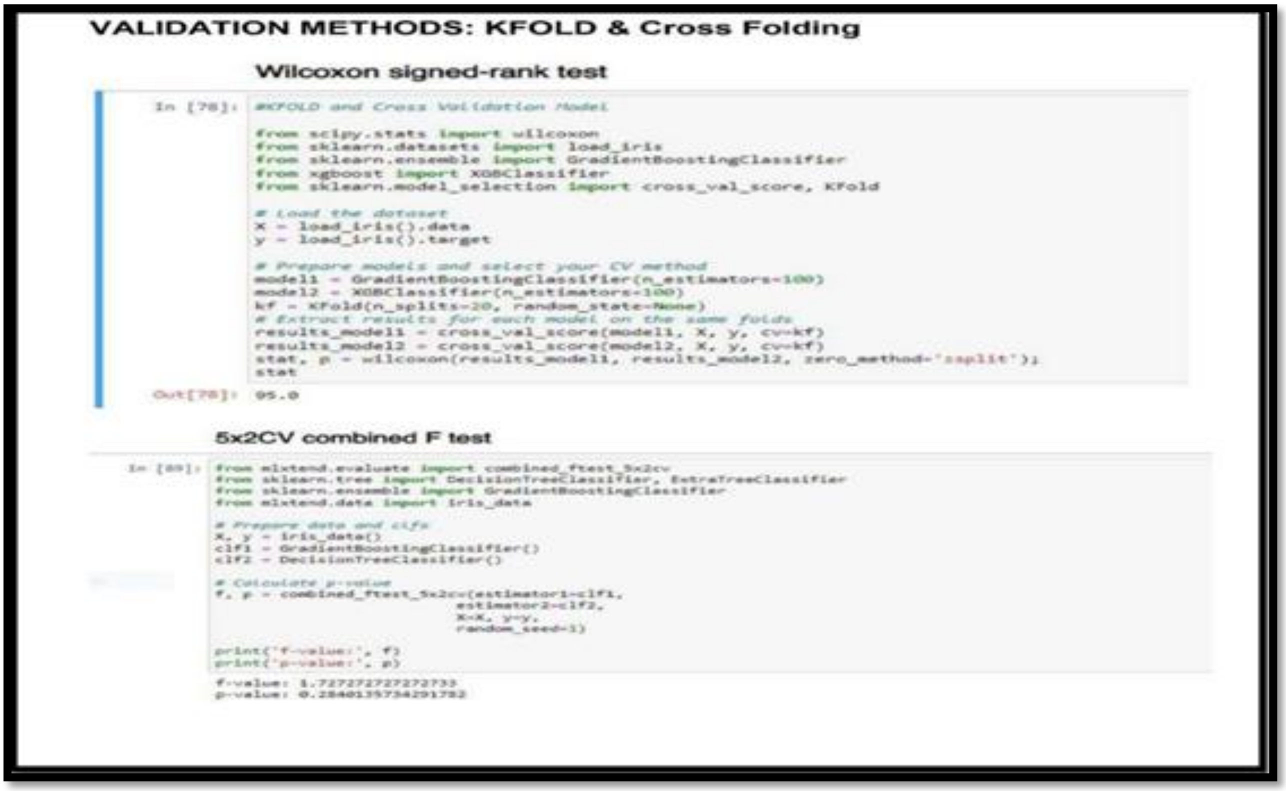
	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956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



2. TUNE THE MODEL – HYPERPARAMETER TUNING

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

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

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

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



VALIDATION METHODS: KFOLD & Cross Folding

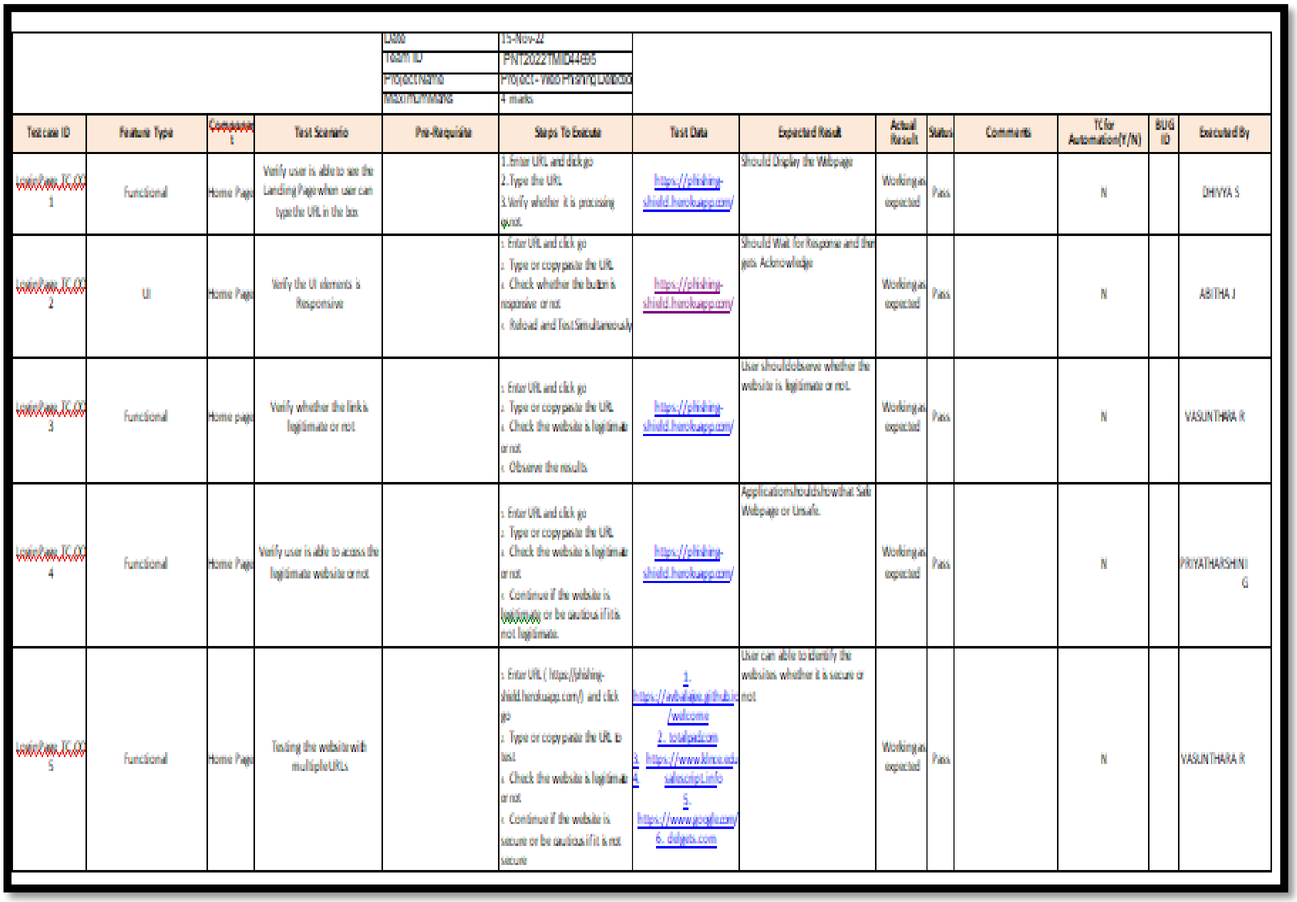
Wilcoxon signed-rank test

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

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

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

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

Out[78]: 95.0
```

5x2CV combined F test

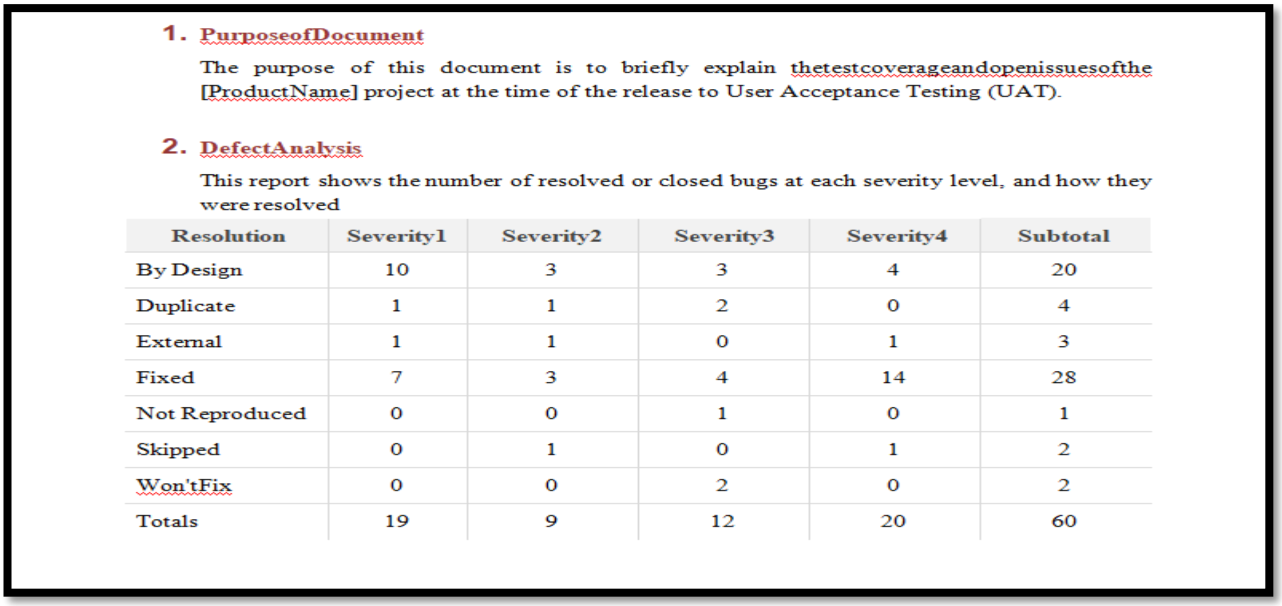
```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
         from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
         from sklearn.ensemble import GradientBoostingClassifier
         from mlxtend.data import iris_data

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

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

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

         f-value: 1.7272727272727273
         p-value: 0.2840135734291782
```

8.2 USER ACCEPTANCE TESTING



Date	15-Nov-22
Team ID	PNT2022TMID44695
Project Name	Project - Web Phishing Detectio
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go 2.Type the URL 3.Verify whether it is processing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		DHIVYA S
LoginPage_TC_OO 2	UI	Home Page	Verify the UI elements is Responsive		1. Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		ABITHA J
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		VASUNTHARA R
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://phishing-shield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		PRIYATHARSHINI G
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs		1. Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimate or not 4. Continue if the website is secure or be cautious if it is not secure	1. https://avbalajee.github.io/welcome 2. totalpad.com 3. https://www.kinco.edu 4. salescript.info 5. https://www.google.com/ 6. delgets.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		VASUNTHARA R



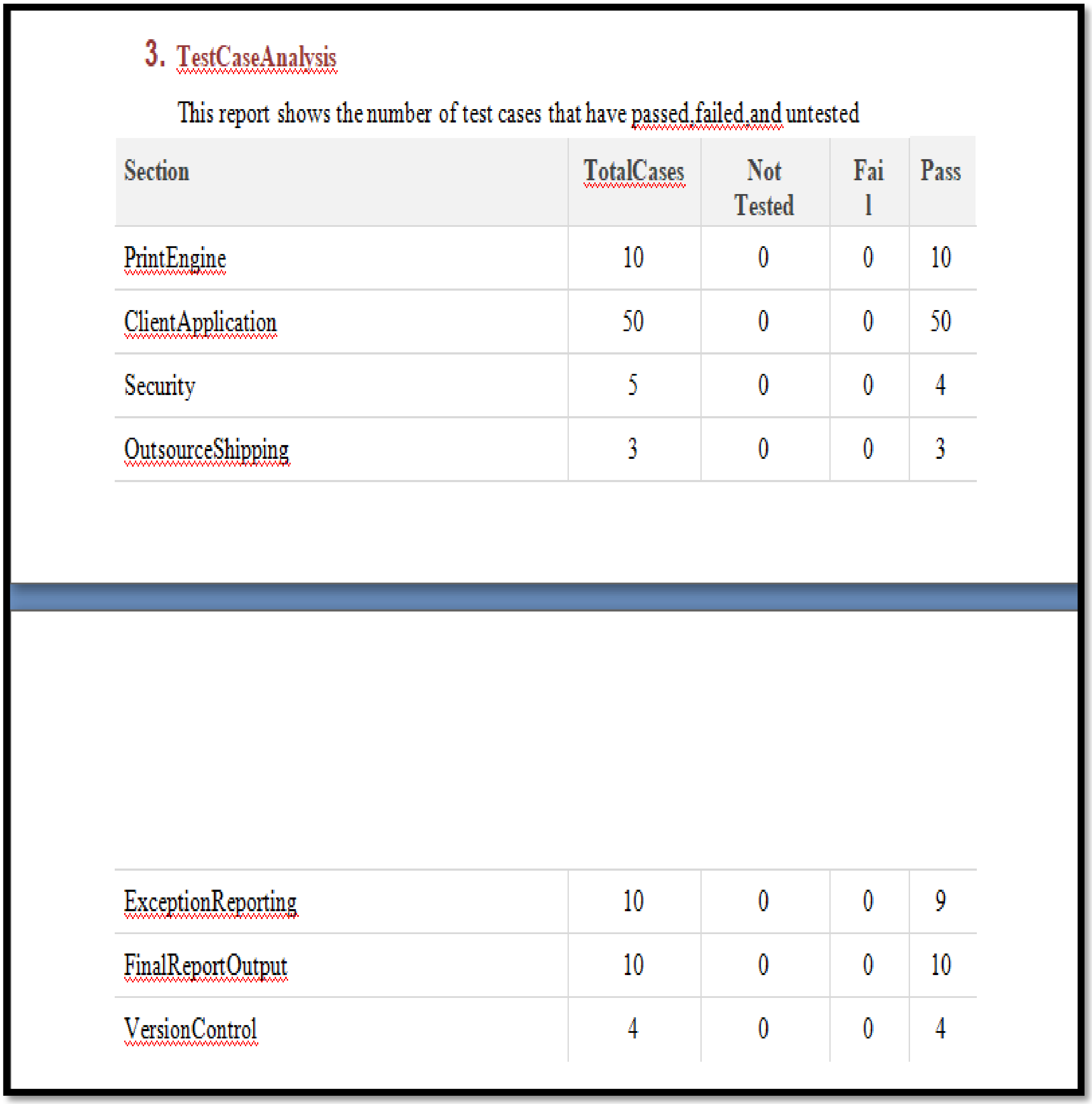
1. PurposeofDocument

The purpose of this document is to briefly explain thetestcoverageandopenissuesofthe [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. DefectAnalysis

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

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	10	3	3	4	20
Duplicate	1	1	2	0	4
External	1	1	0	1	3
Fixed	7	3	4	14	28
Not Reproduced	0	0	1	0	1
Skipped	0	1	0	1	2
Won'tFix	0	0	2	0	2
Totals	19	9	12	20	60



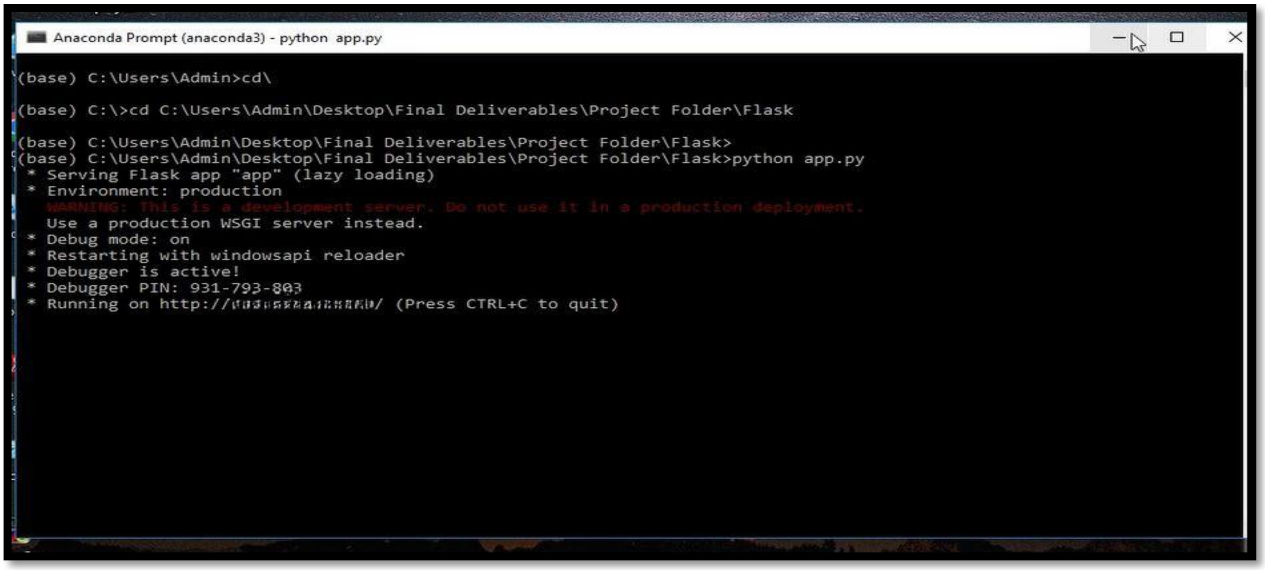
3. TestCaseAnalysis

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

Section	TotalCases	Not Tested	Fail	Pass
PrintEngine	10	0	0	10
ClientApplication	50	0	0	50
Security	5	0	0	4
OutsourceShipping	3	0	0	3
ExceptionReporting	10	0	0	9
FinalReportOutput	10	0	0	10
VersionControl	4	0	0	4

9. RESULTS

9.1 PERFORMANCE METRICS

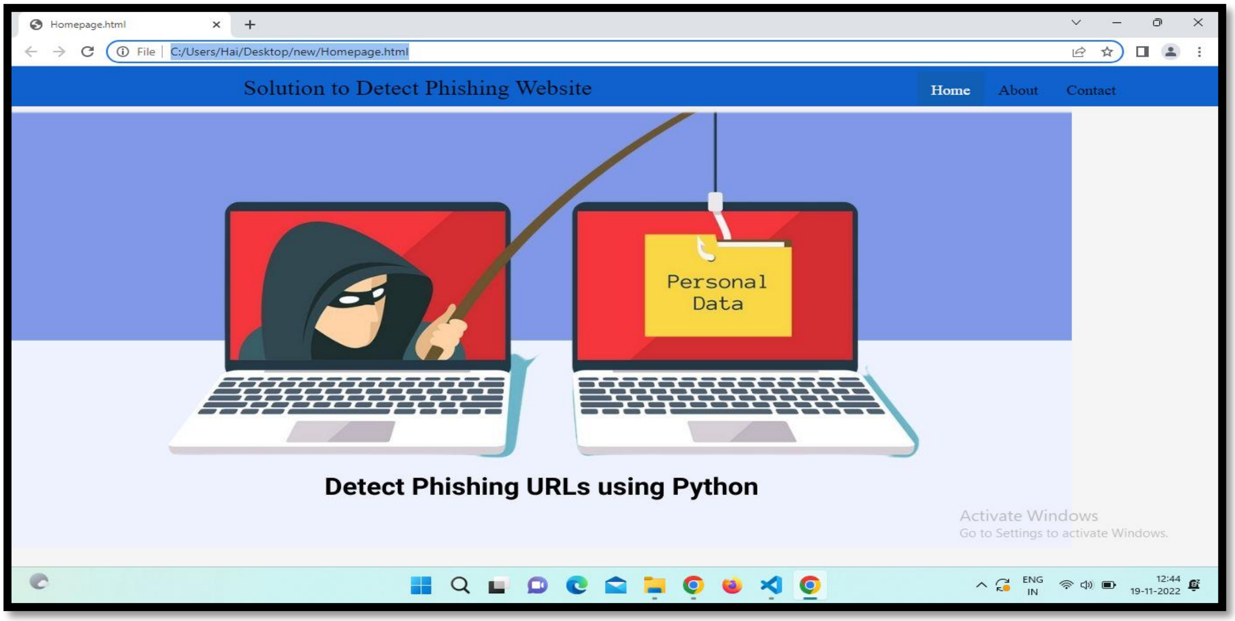


```
(base) C:\Users\Admin>cd\

(base) C:\>cd C:\Users\Admin\Desktop\Final Deliverables\Project Folder\Flask

(base) C:\Users\Admin\Desktop\Final Deliverables\Project Folder\Flask>
(base) C:\Users\Admin\Desktop\Final Deliverables\Project Folder\Flask>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 931-793-803
 * Running on http://[illegible]/ (Press CTRL+C to quit)
```

9.2 OUTPUTS

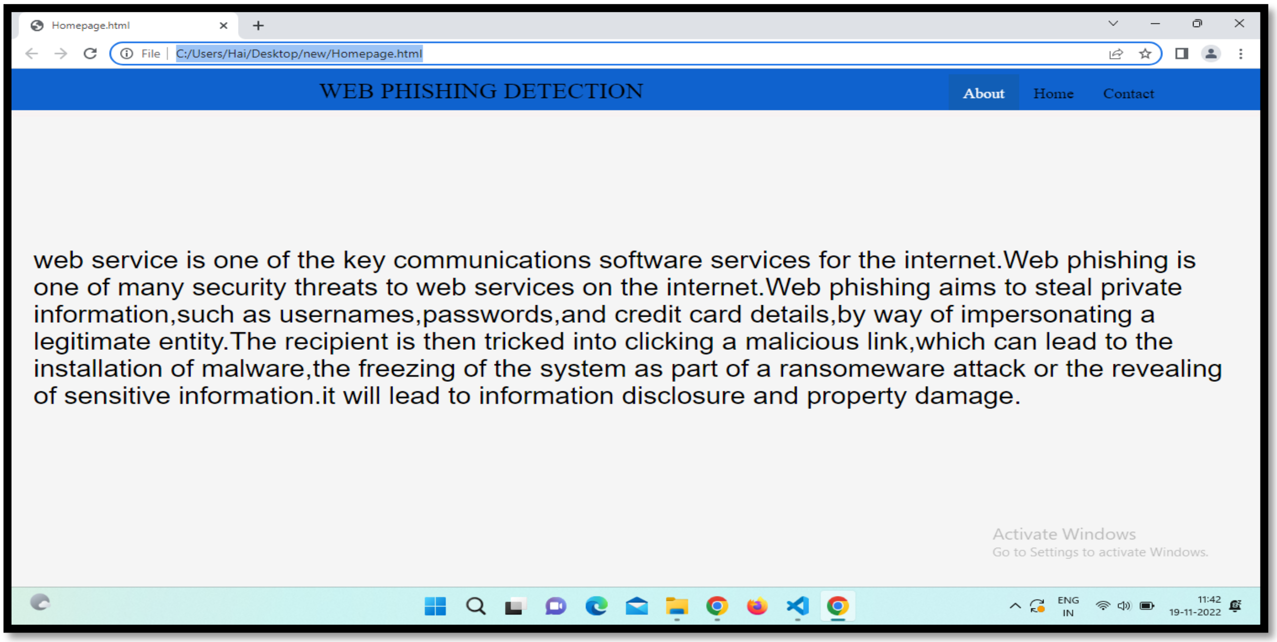
9.2.1 TEASTING THE MODEL

HOME PAGE.HTML



Homepage.html
File
C:/Users/Hai/Desktop/new/Homepage.html
Solution to Detect Phishing Website
Home
About
Contact
Personal Data
Detect Phishing URLs using Python
Activate Windows
Go to Settings to activate Windows.
ENG IN
12:44
19-11-2022

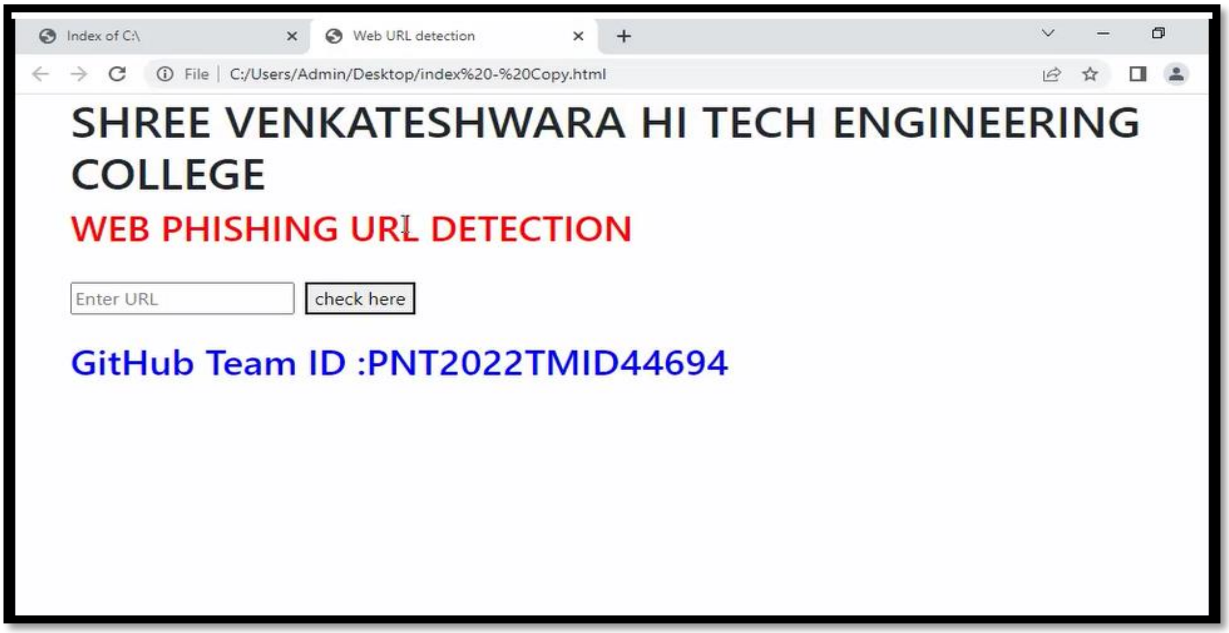
ABOUT PAGE .HTML


Homepage.html
File
C:/Users/Hai/Desktop/new/Homepage.html
About
Home
Contact
Activate Windows
Go to Settings to activate Windows.
ENG
IN
11:42
19-11-2022

WEB PHISHING DETECTION

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.Web phishing aims to steal private information,such as usernames,passwords,and credit card details,by way of impersonating a legitimate entity.The recipient is then tricked into clicking a malicious link,which can lead to the installation of malware,the freezing of the system as part of a ransomeware attack or the revealing of sensitive information.it will lead to information disclosure and property damage.

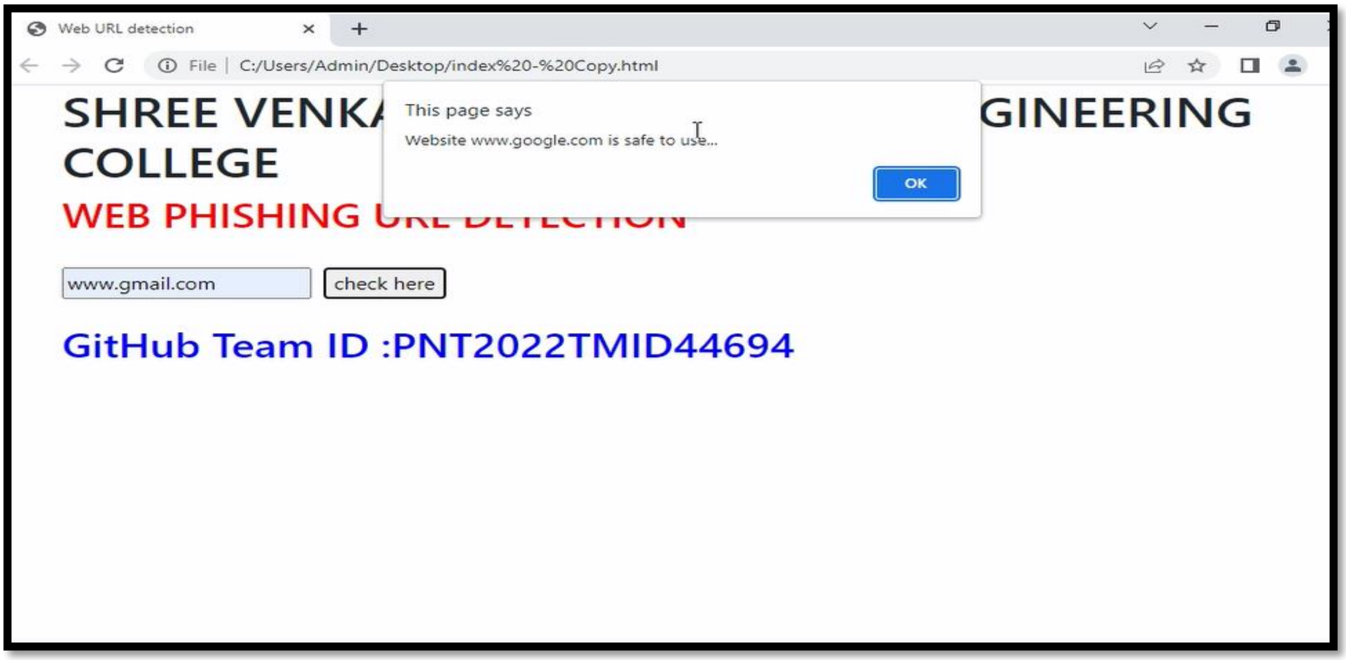
9.2.2 THE FINAL STEP



SHREE VENKATESHWARA HI TECH ENGINEERING COLLEGE

WEB PHISHING URL DETECTION

GitHub Team ID :PNT2022TMID44694



SHREE VENKA GINEERING COLLEGE

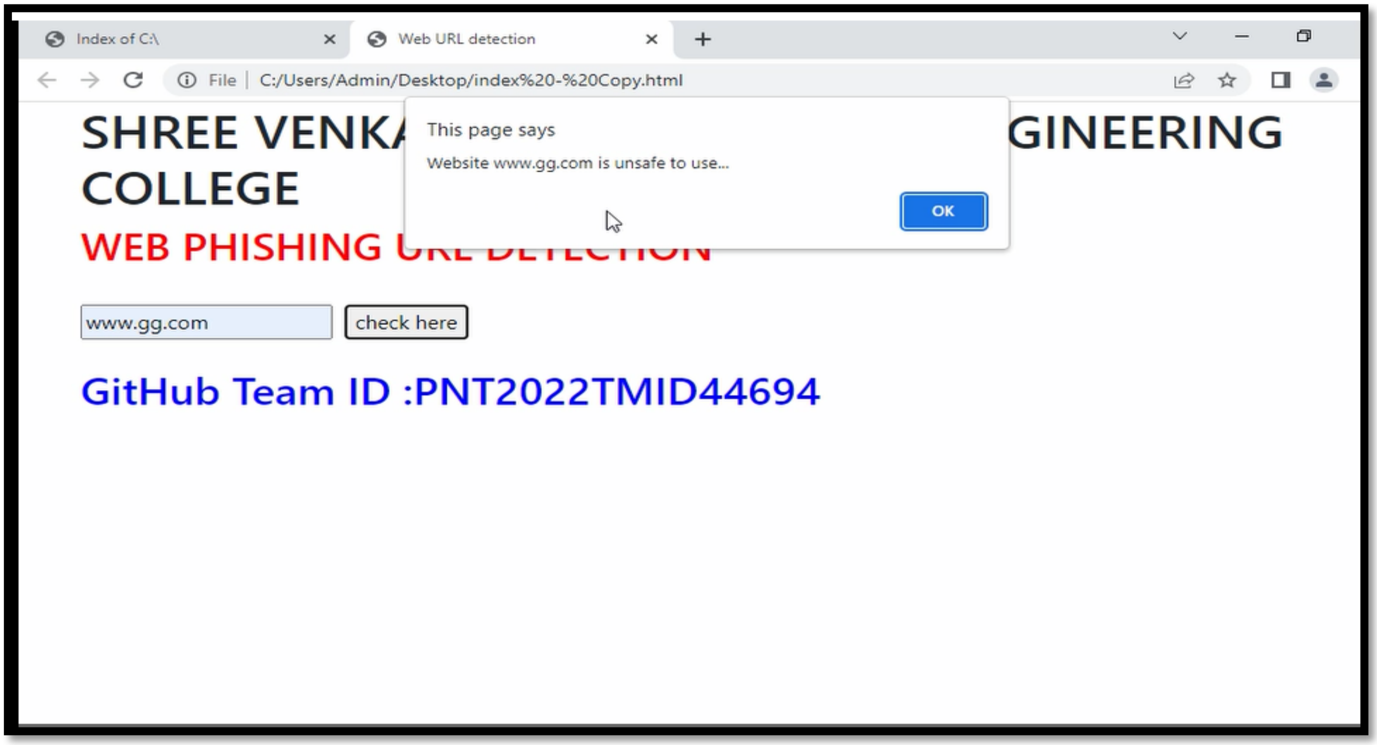
WEB PHISHING U

This page says

Website www.google.com is safe to use...

www.gmail.com

GitHub Team ID :PNT2022TMID44694



SHREE VENKA GINEERING COLLEGE

WEB PHISHING U

This page says

Website www.gg.com is unsafe to use...

www.gg.com

GitHub Team ID :PNT2022TMID44694

10. ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

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

10.2 DISADVANTAGES

- Time consuming .
- huge number of features Agent (MTA) and mail user.
- consuming memory.
- Non standard classifier.
- Time consuming because this technique has many layers to make the final result.
- huge number of features many algorithm for classification which mean time consuming
- higher cost .
- need large mail server and high memory requirement.
- Less accuracy because it depend on unsupervised learning, need feed continuously.
- Need feed continuously.

11. CONCLUSION

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

12.FUTURE SCOPE

The means by which hackers access user information have quickly evolved beyond traditional phishing emails. Phishing has always had the aim of baiting users to take an action or share a piece of sensitive information by appearing as a non-threat but awareness has since grown. Unprompted password reset emails, while once effective, no longer drive the same volume of user action and are often detected by spam filters.

Today, phishing attacks are targeted, can be difficult to detect, and grant malicious individuals broad permissions over user data, user devices, and online services. The days of basic phishing schemes have more or less passed. Attacks now rely on advanced forms of infiltration that better disguise malicious intent.

13. APPENDIX

13.1 SOURCE CODE

HOME PAGE.HTML

```
<html>
<script>
</script>
<style>
.header { position: relative;
top:0;
margin:0px;
z-index: 1;
left: 0px;
right: 0px;
position: fixed;
background-color: #0f62ce ;
color: white;
box-shadow: 0px 8px 2px #f7f3f3f6;
overflow: hidden;
padding-left:20px;
font-family: 'Times New Roman';
font-size: 2vw;
width: 100%;
height:8%;
text-align: center;
}
.topnav {
overflow: hidden;
background-color: #1446b3;
```

```
}
.topnav-right a {
float: left;
color: black;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 18px;
}
.topnav-right a:hover {
background-color: #1846dd;
color: black;
}
.topnav-right a.active {
background-color: #115eb6;
color: white;
}
.topnav-right {
float: right;
padding-right:100px;
}
body {
background-image: -webkit-linear-gradient(90deg, skyblue 0%, steelblue 100%);
background-image: url("");
background-size: cover;
background-attachment: fixed;
background-size: 100% 100%;
background-color: ;
background-repeat: no-repeat;
background-size:cover;
background-position: 0px 0px;
```

```
}
.button {
background-color: #091425;
border: none;
color: white;
padding: 15px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 12px;
border-radius: 16px;
}
.button:hover {
box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
display: inline-block;
margin-bottom:18px;
border: 1px solid #ccc;
box-sizing: border-box;
}
button {
background-color: #091425;
color: white;
padding: 14px 20px;
margin-bottom:10px;
border: none;
cursor: pointer;
```

```
width: 17%;
border-radius:4px;
font-family:Montserrat;
}
button:hover {
opacity: 0.8;
}
.cancelbtn {
width: auto;
padding: 10px 18px;
background-color: #f44336;
}
.imgcontainer {
text-align: center;
margin: 24px 0 12px 0;
}
img.avatar {
width: 30%;
border-radius: 50%;
}
.container {
padding: 16px;
}
span.psw {
float: right;
padding-top: 16px;
}
/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
span.psw {
display: block;
float: none;
}
.cancelbtn {
width: 100%;
}
}
.home{
margin:80px;
width: 84%;
height: 500px;
padding-top:10px;
padding-left: 30px;
}
.login{
margin:80px;
box-sizing: content-box;
width: 84%;
height: 420px;
padding: 30px;
border: 10px solid blue;
}
.left,.right{
box-sizing: content-box;
height: 400px;
margin:20px;
border: 10px solid blue;
}
.mySlides {display: none;}
img {vertical-align: middle;}
/* Slideshow container */
.slideshow-container {
```

```
max-width: 1000px;
position: relative;
margin: auto;
}
/* Caption text */
.text {
color: #f2f2f2;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
width: 100%;
text-align: center;
}
/* The dots/bullets/indicators */
.dot {
height: 15px;
width: 15px;
margin: 0 2px;
background-color: #bbb;
border-radius: 50%;
display: inline-block;
transition: background-color 0.6s ease;
}
.active {
background-color: #FCAD98;
}
/* Fading animation */
.fade {
-webkit-animation-name: fade;
-webkit-animation-duration: 1.5s;
```

```
animation-name: fade;
animation-duration: 1.5s;
}
@-webkit-keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
@keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
.text {font-size: 11px}
}
@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');
* {
box-sizing: border-box;
}
body {
min-height: 100vh;
margin: 0;
color: #fff;
font-family: 'Poppins',sans-serif;
display: flex;
align-items: center;
justify-content: center;
background-color: #f5f5f5;
}
.container {
max-width: 1376px;
```

```
margin: auto;
padding: 2rem 1.5rem;
}
.cards {
display: flex;
flex-wrap: wrap;
align-items: center;
justify-content: center;
}
.card {
cursor: pointer;
background-color: transparent;
height: 300px;
perspective: 1000px;
margin: 1rem;
align-items: center;
justify-content: center;
}
.card h3 {
border-bottom: 1px #fff solid;
padding-bottom: 10px;
margin-bottom: 10px;
text-align: center;
font-size: 1.6rem;
word-spacing: 3px;
}
.card p{
opacity: 0.75;
font-size: 0.8rem;
line-height: 1.4;
}
```

```
.card img {
width: 360px;
height: 300px;
object-fit: cover;
border-radius: 3px;
}
.card-inner {
position: relative;
width: 360px;
height: 100%;
transition: transform 0.9s;
transform-style: preserve-3d;
}
.card:hover .card-inner {
transform: rotateY(180deg);
}
.card-front,
.card-back {
position: absolute;
width: 360px;
height: 100%;
-webkit-backface-visibility: hidden;
backface-visibility: hidden;
}
.card-back {
background-color: #222;
color: #fff;
padding: 1.5rem;
transform: rotateY(180deg);
}
.text-block {
```

```
position: absolute;
bottom: 20px;
right: 20px;
background-color: black;
color: white;
padding-left: 20px;
padding-right: 20px;
}
p
{
color:black;
font-style:Times New Roman;
font-size:30px;
}
</style>
<body style="background-image:url({{url_for('static',file')}});background-position:
center;background-repeat: no-repeat;
background-size: cover;">
 <img class="image image-contain" src="https://cdn.activestate.com/wp-
content/uploads/2021/02/phishing-detection-with-Python.jpg"alt="MDN logo" />

<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:black; padding-
top:1%;padding-left:18%;">Solution to Detect Phishing Website</div>
<div class="topnav-right"style="padding-top:0.5%;">
<a class="active" href="/home">Home</a>
<a href="/about">About</a>
<a href="/contact">Contact</a>
</div>
</div>
<div class="container">
```

```
</div>
</body>
<html>
```

ABOUT.HTML

```
<html>
<script>
</script>
<style>
.header { position: relative;
top:0;
margin:0px;
z-index: 1;
left: 0px;
right: 0px;
position: fixed;
background-color: #0f62ce ;
color: white;
box-shadow: 0px 8px 2px #f7f3f3f6;
overflow: hidden;
padding-left:20px;
font-family: 'Times New Roman';
font-size: 2vw;
width: 100%;
height:8%;
text-align: center;
}
.topnav {
overflow: hidden;
```

```
background-color: #1446b3;
}
.topnav-right a {
float: left;
color: black;
text-align: center;
padding: 14px 16px;
text-decoration: none;
font-size: 18px;
}
.topnav-right a:hover {
background-color: #1846dd;
color: black;
}
.topnav-right a.active {
background-color: #115eb6;
color: white;
}
.topnav-right {
float: right;
padding-right:100px;
}
body {
background-image: -webkit-linear-gradient(90deg, skyblue 0%, steelblue 100%);
background-image: url("");
background-size: cover;
background-attachment: fixed;
background-size: 100% 100%;
background-color: ;
background-repeat: no-repeat;
background-size:cover;
```

```
background-position: 0px 0px;
}
.button {
background-color: #091425;
border: none;
color: white;
padding: 15px 32px;
text-align: center;
text-decoration: none;
display: inline-block;
font-size: 12px;
border-radius: 16px;
}
.button:hover {
box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
display: inline-block;
margin-bottom:18px;
border: 1px solid #ccc;
box-sizing: border-box;
}
button {
background-color: #091425;
color: white;
padding: 14px 20px;
margin-bottom:10px;
border: none;
```

```
cursor: pointer;
width: 17%;
border-radius:4px;
font-family:Montserrat;
}
button:hover {
opacity: 0.8;
}
.cancelbtn {
width: auto;
padding: 10px 18px;
background-color: #f44336;
}
.imgcontainer {
text-align: center;
margin: 24px 0 12px 0;
}
img.avatar {
width: 30%;
border-radius: 50%;
}
.container {
padding: 16px;
}
span.psw {
float: right;
padding-top: 16px;
}
/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
span.psw {
```

```
display: block;
float: none;
}
.cancelbtn {
width: 100%;
}
}
.home{
margin:80px;
width: 84%;
height: 500px;
padding-top:10px;
padding-left: 30px;
}
.login{
margin:80px;
box-sizing: content-box;
width: 84%;
height: 420px;
padding: 30px;
border: 10px solid blue;
}
.left,.right{
box-sizing: content-box;
height: 400px;
margin:20px;
border: 10px solid blue;
}
.mySlides {display: none;}
img {vertical-align: middle;}
/* Slideshow container */
```

```
.slideshow-container {
max-width: 1000px;
position: relative;
margin: auto;
}
/* Caption text */
.text {
color: #f2f2f2;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
width: 100%;
text-align: center;
}
/* The dots/bullets/indicators */
.dot {
height: 15px;
width: 15px;
margin: 0 2px;
background-color: #bbb;
border-radius: 50%;
display: inline-block;
transition: background-color 0.6s ease;
}
.active {
background-color: #FCAD98;
}
/* Fading animation */
.fade {
-webkit-animation-name: fade;
```

```
-webkit-animation-duration: 1.5s;
animation-name: fade;
animation-duration: 1.5s;
}
@-webkit-keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
@keyframes fade {
from {opacity: .4}
to {opacity: 1}
}
/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
.text {font-size: 11px}
}
@import url('https://fonts.googleapis.com/css2?family=Poppins&display=swap');
* {
box-sizing: border-box;
}
body {
min-height: 100vh;
margin: 0;
color: #fff;
font-family: 'Poppins',sans-serif;
display: flex;
align-items: center;
justify-content: center;
background-color: #f5f5f5;
}
.container {
```

```
max-width: 1376px;
margin: auto;
padding: 2rem 1.5rem;
}
.cards {
display: flex;
flex-wrap: wrap;
align-items: center;
justify-content: center;
}
.card {
cursor: pointer;
background-color: transparent;
height: 300px;
perspective: 1000px;
margin: 1rem;
align-items: center;
justify-content: center;
}
.card h3 {
border-bottom: 1px #fff solid;
padding-bottom: 10px;
margin-bottom: 10px;
text-align: center;
font-size: 1.6rem;
word-spacing: 3px;
}
.card p{
opacity: 0.75;
font-size: 0.8rem;
line-height: 1.4;
```

```
}
.card img {
width: 360px;
height: 300px;
object-fit: cover;
border-radius: 3px;
}
.card-inner {
position: relative;
width: 360px;
height: 100%;
transition: transform 0.9s;
transform-style: preserve-3d;
}
.card:hover .card-inner {
transform: rotateY(180deg);
}
.card-front,
.card-back {
position: absolute;
width: 360px;
height: 100%;
-webkit-backface-visibility: hidden;
backface-visibility: hidden;
}
.card-back {
background-color: #222;
color: #fff;
padding: 1.5rem;
transform: rotateY(180deg);
}
```

```
.text-block {
position: absolute;
bottom: 20px;
right: 20px;
background-color: black;
color: white;
padding-left: 20px;
padding-right: 20px;
}
p
{
color:black;
font-style:Times New Roman;
font-size:30px;
}
</style>
<body style="background-
image:url({{url_for('static',filename='images/bck3.png')}});background-position:
center;background-repeat: no-repeat;
background-size: cover;">
<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:black; padding-
top:1%;padding-left:23.5%;">WEB PHISHING DETECTION</div>
<div class="topnav-right"style="padding-top:0.5%;">
<a class="active" href="/about">About</a>
<a href="/home">Home</a>
<a href="/contact">Contact</a>
</div>
</div>
<div class="container">
```

```
<p>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.Web phishing aims to steal private information,such as usernames,passwords,and credit card details,by way of impersonating a legitimate entity.The recipient is then tricked into clicking a malicious link,which can lead to the installation of malware,the freezing of the system as part of a ransomeware attack or the revealing of sensitive information.it will lead to information disclosure and property damage.</p>
</div>
</body>
<html>
```

APP.PY

```
#importing required libraries

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

file = open("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)
```

FEATURE

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
```

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

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

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

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

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

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


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

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
```

```
        self.features.append(self.DNSRecording())
        self.features.append(self.WebsiteTraffic())
        self.features.append(self.PageRank())
        self.features.append(self.GoogleIndex())
        self.features.append(self.LinksPointingToPage())
        self.features.append(self.StatsReport())


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

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

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

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

```
                    'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                    'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

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

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

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

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

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

```
        return -1

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

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

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

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

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

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

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

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

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

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

        for embed in self.soup.find_all('embed', src=True):
            dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
```

```
                if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

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

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

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

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

    except:
        return -1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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

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

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.l
y', url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match                                                                                     =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.
158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.
151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'

'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.
224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.1
9\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'

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

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

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

IBM 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

import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "V7p-6PyIwKuNRoU4L2a38uPSAmc8QzciVBPNZbhI2hne"
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}


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)

        # NOTE: manually define and pass the array(s) of values to be scored in the next line
        payload_scoring = {"input_data": [{"fields":
[['f0','f1','f2','f3','f4','f5','f6','f7','f8','f9','f10','f11','f12','f13','f14','f15','f16','f17','f18','f19','f20','f21','f
22','f23','f24','f25','f26','f27','f28','f29']], "values": [[-1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,1,-1,1,-1,-1,-
1,0,1,1,1,1,-1,-1,-1,-1,1,1,-1]]}]}

        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/7bc163a4-37f7-4ec6-b0bf-
63d1e40a85e4/predictions?version=2022-11-16', json=payload_scoring,
        headers={'Authorization': 'Bearer ' + mltoken})
        #print("Scoring response")
        #print(response_scoring.json())
        pred=response_scoring.json()
        output=pred['predictions'][0]['values'][0][0]

        y_pred =output
        #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 gg{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)

    # NOTE: manually define and pass the array(s) of values to be scored in the next line
    payload_scoring = {"input_data": [{"fields":
[['f0','f1','f2','f3','f4','f5','f6','f7','f8','f9','f10','f11','f12','f13','f14','f15','f16','f17','f18','f19','f20','f21','f
22','f23','f24','f25','f26','f27','f28','f29']], "values": [[-1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,1,-1,1,-1,-1,-
1,0,1,1,1,1,-1,-1,-1,-1,1,1,-1]]}]}

    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/7bc163a4-37f7-4ec6-b0bf-
63d1e40a85e4/predictions?version=2022-11-16', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    #print("Scoring response")
    #print(response_scoring.json())
    pred=response_scoring.json()
    output=pred['predictions'][0]['values'][0][0]
```

INPUTSCRIPT

```
import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime
import requests
import favicon
import re
import google
from googlesearch import search

#checking if URL contains any IP address. Returns -1 if contains else returns 1
def having_IPhaving_IP_Address(url):
    match=regex.search('(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-
5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  #IPv4
                    '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-
F]{1,2})\\/)'  #IPv4 in hexadecimal
                    '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)     #Ipv6
    if match:
    #print match.group()
        return -1
    else:
    #print 'No matching pattern found'
        return 1

#Checking for the URL length. Returns 1 (Legitimate) if the URL length is less than 54
characters
#Returns 0 if the length is between 54 and 75
#Else returns -1;
def URLURL_Length (url):
    length=len(url)
    if(length<=75):
         if(length<54):
             return 1
         else:
             return 0
    else:
        return -1

#Checking with the shortening URLs.
#Returns -1 if any shortening URLs used.
#Else returns 1
def Shortining_Service (url):

match=regex.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|
'
                    'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

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

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

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

#Checking for @ symbol. Returns 1 if no @ symbol found. Else returns 0.
```

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

#Checking for Double Slash redirections. Returns -1 if // found. Else returns 1
def double_slash_redirecting(url):
    for i in range(8,len(url)):
        if(url[i]=='/'):


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

#Checking for - in Domain. Returns -1 if '-' is found else returns 1.
def Prefix_Suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return -1
    else:
        return  1

#checking the Subdomain. Returns 1 if the subDomain contains less than 1 '.'
#Returns 0 if the subDomain contains less than 2 '.'
#Returns -1 if the subDomain contains more than 2 '.'
def having_Sub_Domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')<=2):
        if(subDomain.count('.')<=1):
```

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

#Checking the SSL. Returns 1 if it returns the respomse code and -1 if exceptions are thrown.
def SSLfinal_State(url):
  try:
      response = requests.get(url)
      return 1
  except Exception as e:
      return -1

#domains expires on ≤ 1 year returns -1, otherwise returns 1

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

#Checking the Favicon. Returns 1 if the domain of the favicon image and the URL domain match else returns -1.

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

#Checking the Port of the URL. Returns 1 if the port is available else returns -1.
def port(url):
    try:
        a_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        location=(url[7:],80)
        result_of_check = a_socket.connect_ex(location)
        if result_of_check == 0:
            return 1
        else:
            return -1
        a_socket.close
    except:
        return -1

# HTTPS token in part of domain of URL returns -1, otherwise returns 1
def HTTPS_token(url):
    match=re.search('https://|http://',url)
    if (match.start(0)==0):
        url=url[match.end(0):]
    match=re.search('http|https',url)
    if match:
        return -1
    else:
        return 1



#% of request URL<22% returns 1, otherwise returns -1
def Request_URL(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)
        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

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

#:% of URL of anchor<31% returns 1, % of URL of anchor ≥ 31% and ≤ 67% returns 0, otherwise returns -1
def URL_of_Anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

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

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

#:% of links in <meta>, <script>and<link>tags < 25% returns 1, % of links in <meta>,
#<script> and <link> tags ≥ 25% and ≤ 81% returns 0, otherwise returns -1

def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

#Server Form Handling
#SFH is "about: blank" or empty → phishing, SFH refers to a different domain → suspicious, otherwise → legitimate
def SFH(url):
    #ongoing
    return -1

#:using "mail()" or "mailto:" returning -1, otherwise returns 1
def Submitting_to_email(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:','mail():')):
            return -1
        else:
            return 1
    except:
        return -1

#Host name is not in URL returns -1, otherwise returns 1
def Abnormal_URL(url):
    subDomain, domain, suffix = extract(url)
    try:
        domain = whois.whois(url)
        hostname=domain.domain_name[0].lower()
        match=re.search(hostname,url)
        if match:
            return 1
        else:
            return -1
    except:
        return -1

#number of redirect page ≤ 1 returns 1, otherwise returns 0
def Redirect(url):
    try:
        request = requests.get(url)
        a=request.history
        if(len(a)<=1):
            return 1
        else:
            return 0
    except:
        return 0



#onMouseOver changes status bar returns -1, otherwise returns 1
def on_mouseover(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

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

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

#popup window contains text field → phishing, otherwise → legitimate
def popUpWidnow(url):
    #ongoing
    return 1

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



#:age of domain ≥ 6 months returns 1, otherwise returns -1
def age_of_domain(url):
    try:
        w = whois.whois(url).creation_date[0].year
        if(w<=2018):
            return 1
        else:
            return -1
    except Exception as e:
        return -1

#no DNS record for domain returns -1, otherwise returns 1
def DNSRecord(url):

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

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

#website rank < 100.000 returns 1, website rank > 100.000 returns 0, otherwise returns -1
def web_traffic(url):
    try:
        rank                                                                              =
BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="       +
url).read(), "xml").find("REACH")['RANK']
    except TypeError:
        return -1
    rank= int(rank)
    if (rank<100000):
        return 1
    else:
        return 0
```

```
#:PageRank < 0,2 → phishing, otherwise → legitimate
def Page_Rank(url):
    #ongoing
    return 1

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



#:number of links pointing to webpage = 0 returns 1, number of links pointing to webpage> 0
#and ≤ 2 returns 0, otherwise returns -1

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

#:host in top 10 phishing IPs or domains returns -1, otherwise returns 1
def Statistical_report (url):
    hostname = url
    h = [(x.start(0), x.end(0)) for x in
regex.finditer('https://|http://|www.|https://www.|http://www.', hostname)]
    z = int(len(h))
    if z != 0:
        y = h[0][1]
        hostname = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in regex.finditer('/', hostname)]
        z = int(len(h))
        if z != 0:
            hostname = hostname[:h[0][0]]

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

ip_match=regex.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.11
6|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\.1
```

```
08|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\
.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|1
0\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.1
04\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|2
08\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217
\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.2
14\.197\.72|87\.98\.255\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.
46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',
ip_address)
    except:
        return -1

    if url_match:
        return -1
    else:
        return 1

#returning scrapped data to calling function in app.py
def main(url):



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

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

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

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abnormal_URL(url
),
            Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(url),

age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Google_Index(url),
            Links_pointing_to_page(url),Statistical_report(url)]]


    print(check)
    return check
```

PHISHIG DETECTION

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.model_selection import train_test_split
import feature_extraction
def getResult(url):
#Importing dataset
    data= np.loadtxt("dataset_website.csv", delimiter = "")
#Seperating features and labels
    X = data[:, -1]
    y = data[:, -1]
#Seperating training features, testing features, training labels & testing labels
x_train, x_test, y_train, y_test,train_test_split(x, y, test_size = 0.2)
clf- rfc()
clf.fit(x_train, y_train) "score" clf.score(x_test, y_test)
print(score 100)
X_new = []
X input url
X_new-feature extraction.generate_data_set(X_input)
X new np.array(X_new).reshape(1,-1)
```

```
try:
    prediction clf.predict(X_new)
if prediction -1:
    return "Phishing Url"
else:
    return "Legitimate Url"
except:
    return "Phishing Url"
```

FINAL.HTML

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

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

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

```
</head>

<body bgcolor="red">

<div class=" container">
    <div class="row">
        <div class="form col-md" id="form1">
            <h1>SHREE VENKATESHWARA HI TECH ENGINEERING COLLEGE</h1>
            <h2><font color="red">WEB PHISHING URL DETECTION</font></h2>

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

                           <button onclick="myFunction()">check here</button>




    </div>



</div>
<br>
<h2><font color="blue">GitHub Team ID :PNT2022TMID44694</font></h2>

</div>
```

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


    <script>

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

    </script>

</body>

</html>
```

STYLE CSS:

```
*,
*::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
  font-size: 62,5%;
}

body {
  padding: 10% 5%;
  background: #0f2027;
  background: linear-gradient(to right,#2c5364, #203a43, #0f2027);
  justify-content: center;
  align-items: center;
  height: 100vh;
  color: #fff;
}
```

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

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

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

```
.button {
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, #fff, #f8eedb);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
}
```

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

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


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

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

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

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

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

}

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-42766-1660708741

PROJECT DEMO LINK

https://drive.google.com/file/d/1UpwWkd2QUo_gYL2mJ-yD_dWcGrzJmDPX/view?usp=sharing