IBM
NALAIYA THIRAN

PROJECT REPORT
ON
WEB PHISHING DETECTION

TEAM ID: PNT2022TMID15315

R.M.D.ENGINEERING COLLEGE

Team Members

SIDDAREDDY TEJASWI REDDY
SARVUGARI ANUSHA
SHALINI A
SIVA SRUTHI KODURU

TABLE OF CONTENTS

ABSTRACT

Phishing is the most commonly used social engineering and cyber attack. Through such attacks, the phisher targets naive online users by tricking them into revealing confidential information, with the purpose of using it fraudulently. In order to avoid getting phished, Users should have awareness of phishing websites. Have a blacklist of phishing websites which requires the knowledge of website being detected as phishing. Detect them in their early appearance, using machine learning and deep neural network algorithms. Of the above three, the machine learning based method is provento be most effective than the other methods. A phishing website is a common social engineering method that mimics trustful uniform resource locators (URLs) and webpages. The objective of this project is to train machine learning models and deep neural nets on the dataset created to predict phishing websites. Both phishing and benign URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted. The performance level of each model is measured and compared.

Keywords: Deep learning, Machine learning, Phishing website attack, Phishing website detection, Anti-phishing website, Legitimate website , Phishing website datasets, Phishing website features.

PRE-REQUISITES

TOOLS : JUPITER NOTEBOOK

OPERATING SYSTEM : WINDOWS 11

LANGUAGE : PYTHON

INSTALLING LIBRARIES

In this first step, we have to import the most common libraries used in python for machine learning such as

- Pandas
- Numpy
- Seaborn
- Matplotlib

IMPORTING DATA

In this project, we have used the url pre processed data.

CHAPTER 1

INTRODUCTION

Phishing imitates the characteristics and alternatives of emails and makes it appear similar due to the fact the original one. It seems nearly like that of the legitimate supply. The consumer thinks that this e-mail has come back from a real employer or a corporation. This makes the consumer to forcefully visit the phishing internet site thru the hyperlinks given inside the phishing email. These phishing web sites region unit created to mock the seams of an ingenious website. The phishers force person to inventory up the non-public info via giving baleful messages or validate account messages etc. so that they inventory up the preferred data which might be utilized by them to misuse it. They devise things such as the user isn't always left with the other choice but to go to their spoofed web site. Phishing is the most hazardous criminal physical activities in the cyber region. Since the maximum of the customers logs on to get admission to the services supplied with the aid of government and financial establishments, there has been a significant boom in phishing attacks for the beyond few years. Phishers commenced to earn cash and that they try this as a thriving business.

Phishing may be law-breaking, the explanation behind the phishers doing this crime is that it is terribly trustworthy to try to do this, it doesn't value something and it effective. The phishing will truly get entry to the e-mail identity of somebody it's terribly sincere to are looking for out the email identification currently every day and you will send an email to every person is freely offered throughout the globe. These attacker’s vicinity terribly much less price and electricity to urge valuable know-how quick and truly. The phishing frauds effects malware infections, statistics loss, fraud, etc. information at some stage in which those cyber criminals have an interest is that the crucial data of a user similar to the password, OTP, credit/ debit card numbers CVV, sensitive know- how associated with business, medical understanding, confidential information, etc commonly these criminals conjointly acquire data which may provide them directly get admission to do the social media account their emails.

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

1.1 PROJECT OVERVIEW

- To develop a novel approach to detect malicious URL and alert users.
- To apply ML techniques in the proposed approach in order to analyze the realtime URLs and produce effective results.
- To implement the concept of RNN, which is a familiar ML technique that has the capability to handle huge amount of data.

1.2 PURPOSE

- To develop an unsupervised deep learning method to generate insight from a URL.
- The study can be extended in order to generate an outcome for a larger network and protect the privacy of an individual.

CHAPTER 2

LITERATURE SURVEY

PAPER 2.1: PHISH-SAFE: URL Features-Based Phishing Detection System Using Machine Learning.

Authors: Ankit Kumar Jain **&** B.B.Gupta

Abstract:

Today, phishing is one of the most serious cyber-security threat in which attackers steal sensitive information such as personal identification number(PIN), credit card details, login, password, etc., from Internet users. In this paper, we proposed a machine learning based anti-phishing system (i.e., named as PHISH- SAFE) based on Uniform Resource Locator (URL) features. To evaluate the performance of our proposed system, we have taken 14 features from URL to detect a website as a phishing or non-phishing. The proposed system is trained using more than 33,000 phishing and legitimate URLs with SVM and Naïve Bayes classifiers.

Our experiment results show more than 90% accuracy in detecting phishing websites using SVM classifier.

PAPER 2.2: Detection of URL based phishing attacks using machine learning

Authors: Ms. Sophiya. Shikalgar, Dr. S. D. Sawarkar, Mrs. Swati Narwane

Abstract:

A fraud effort to get sensitive and personal information like password, username, and bank details like credit / debit card details by masking as a reliable organization in electronic communication. It most of the time redirects the users to similar looking website as legitimate website. The phishing website will appear same as the legitimate website and directs the user to a page to enter personal details of the user on the fake website. The system administration is very important these days as any failure can be detected and solved instantly. The system administration also need to define rules and set firewall settings to avoid phishing attacks through URL. Researchers have been studying various machine learning algorithm in lines to predict and avoid phishing attacks. Through machine learning algorithms one can improve the accuracy of the prediction. The machine learning, no one algorithm works best for every problem, and it’s especially relevant for supervised learning. Using a single machine learning algorithm will give us good accuracy to predict the phishing attacks but to get better accuracy we need something more. The proposed

system predicts the URL based phishing attacks with maximum accuracy. We shall talk about various machine learning, the algorithm which can help in decision making and prediction. We shall use more than one algorithm to get better accuracy of prediction. The algorithms namely the Naive Bayes and Random forest are used in the proposed system to detect URL based phishing attacks. The hybrid algorithm approach by combining.

PAPER 2.3: An Ideal Approach for Detection and Prevention of Phishing Attacks
Authors: Narendra.M & Chaithali shah
Abstract:

In this paper, we propose a phishing detection and prevention approach combining URL-based and Webpage similarity based detection. URL-based phishing detection involves extraction of actual URL (to which the website is actually directed) and the visual URL (which is visible to the user). LinkGuard Algorithm is used to analyze the two URLs and finally depending on the result produced by the algorithm the procedure proceeds to the next phase. If phishing is not detected or Phishing possibility is predicted in URL-based detection, the algorithm proceeds to the visual similarity based detection. A novel technique to visually compare a suspicious page with the legitimate oneis presented.

PAPER 2.4: Phishing website detection based on effective machine learning approach
Authors: Lokesh.G & Gowtham.B
Abstract:
Phishing a form of cyber-attack, which has an adverse effect on people where the user is directed to fake websites and duped to reveal their sensitive and personal information which includes passwords of accounts, bank details, atm pin-card details etc. Hence protecting sensitive information from malwares or web phishing is difficult. Machine learning is a study of data analysis and scientific study of algorithms, which has shown results in recent times in opposing phishing pages when distinguished with visualization, legal solutions, including awareness workshops and classic anti-phishing approaches. This paper examines the applicability of ML techniques in identifying phishing attacks and report their positives and negatives. In specific, there are many ML algorithms that have been explored to declare theappropriate choice that serve as anti-phishing tools. We have designed a Phishing Classification system which extracts features that are meant to defeat common phishing detection

approaches. We also make use of numeric representation along with the comparative study of classical machine learningtechniques like Random Forest, K nearest neighbours, Decision Tree, Linear SVC classifier, One class SVM classifier and wrapper-based features selection which contains the metadata of URLs and use the information to determine if a website is legitimate or not.

PAPER 2.5: Machine Learning and Deep Learning Based Phishing Websites Detection: TheCurrent Gaps and Next Directions

Authors: Kibreab Adane & Berhanu Beyene

Abstract:

There are many phishing websites detection techniques in literature, namely white-listing, black-listing, visual-similarity, heuristic-based, and others. However, detecting zero-hour or newly designed phishing website attacks is an inherent property of machine learning and deep learning techniques. By considering a promising solution of machine learning and deep learningtechniques, researchers have made a great deal of effort to tackle the this problem, which persists due to attackers constantly devising novel strategies to exploit vulnerability or gaps in existing anti-phishing measures. In this study, an extensive effort has been made to rigorously review recent studies focusing on Machine Learning and Deep Learning Based Phishing Websites Detection to excavate the root cause of the aforementioned problems and offer suitable solutions. The study followed the significant criterion to search, download, and screen relevant studies, then to evaluate criterion-based selected studies. The findings show that significant research gaps are available in the rigorously reviewed studies. These gaps are mainlyrelated to imbalanced dataset usage, improper selection of dataset source(s), the unjustified reason for using specific train-test dataset split ratio, scientific disputes on website features inclusion and exclusion, lack of universal consensus on phishing website lifespans and on what is defining a small dataset size, and run-time analysis issues.

PAPER 2.6: Detection of phishing websites using an efficient feature-based machine learning framework.

Authors: Royhu Srinivas rao & sathvik

Abstract: In this paper, we propose a novel classification model, based on heuristic features that are extracted from URL, source code, and third-party services to overcome the disadvantages of existing anti-phishing techniques. Our model has been evaluated using eight different machine learning algorithms and out of which, the Random Forest (RF) algorithm performed the best with an accuracy of 99.31%. The experiments were repeated with different (orthogonal and oblique) random forest classifiers to find the best classifier for the phishing website detection. Principal component analysis Random Forest (PCA-RF) performed the best out of all oblique Random Forests (oRFs) with an accuracy of 99.55%. We have also tested our model with the third-party-based features and without third-party-based features to determine the effectiveness of third-party services in the classification of suspicious websites. We also compared our results with the baseline models (CANTINA and CANTINA+).
Our proposed technique outperformed these methods and also detected zero-day.

CHAPTER 3

3.1 EXISTING PROBLEM

In this technological era, the Internet has made its way to become an inevitable part of our lives. It leads to many convenient experiences in our lives regarding communication, entertainment, education, shopping and so on. As we progress into online life, criminals view the Internet as an opportunity to transfer their physical crimes into a virtual environment. The Internet not only provides convenience in various aspects but also has its downsides, for example, the anonymity that the Internet provides to its users. Presently, many types of crimes have been conducted online. Hence, the main focus of our research is phishing. Phishing is a type of cybercrime where the targets are lured or tricked into giving up sensitive information, such as Social Security Number personal identifiable information and passwords. This obtainment of such information is done fraudulently. Given that phishing is a very broad topic, we have decided that this research should specifically focus on phishing websites.

Rao et al. [1] proposed a novel classification approach that use heuristic-based feature extraction approach. In this, they have classified extracted features into three categories such as URL Obfuscation features, Third-Party-based features, Hyperlink-based features. Moreover, proposed technique gives 99.55% accuracy. Drawback of this is that as this model uses third party features, classification of website dependent on speed of third-party services. Also this model is purely depends on the quality and quantity of the training set and Broken links feature extraction has a Volume 3.

Chunlin et al. [2] proposed approach that primarily focus on character frequency

features. In this they have combined statistical analysis of URL with machine learning technique to get result that is more accurate for classification of malicious URLs. Also they have compared six machine-learning algorithms to verify the effectiveness of proposed algorithm which gives 99.7% precision with false positive rate less than 0.4%.
Sudhanshu et al. [3] used association data mining approach. They have proposed rule based classification technique for phishing website detection. They have concluded that association classification algorithm is better than any other algorithms because of their simple rule transformation. They achieved 92.67% accuracy by extracting 16 features but this is not up to mark so proposed algorithm can be enhanced for efficient detection rate.
M. Amaad et al.[4] presented a hybrid model for classification of phishing website. In this paper, proposed model carried out in two phase. In phase 1,they individually perform classification techniques, and select the best three models based on high accuracy and other performance criteria. While in phase 2, they further combined each individual model with best three model and makes hybrid model that gives better accuracy than individual model. They achieved 97.75% accuracy on testing dataset. There is limitation of this model that it requires more time to build hybrid model.
Hossein et al.[5] developed an open-source framework known as “Fresh-Phish”. For phishing websites, machine-learning data can be created using this framework. In this, they have used reduced features set and using python for building query . They build a large labelled dataset and analyze several machine-learning classifiers against this dataset .Analysis of this gives very good accuracy using machine-learning classifiers. These analyses how long time it takes to train the model.
Gupta et al. [6] proposed a novel anti phishing approach that extracts features from client-side only. Proposed approach is fast and reliable as it is not dependent on third party but it extracts features only from URL and source code. In this paper, they have achieved 99.09% of overall detection accuracy for phishing website. This paper have concluded that this approach has limitation as it can detect webpage written in HTML .Non-HTML webpage cannot detect by this approach.
Bhagyashree et al.[7] proposed a feature based approach to classify URLs as phishing and non phishing. Various features this approach uses are lexical features, WHOIS features, Page Rank and Alexa rank and Phish Tank-based features for disguising phishing and non-phishing website. In this paper, web-mining classification is used.
Mustafa et al.[8] developed safer framework for detecting phishing website. They have extracted URL features of website and using subset based selection technique to obtain better accuracy .In this paper, author evaluated CFS subset based and content based subset selection methods And Machine learning algorithms are used for classification purpose.
Priyanka et al.[9] proposed novel approach by combining two or more algorithms. In this paper ,author has implemented two algorithm Adaline and Backpropion along with SVM for getting good detection rate and classification purpose.
Pradeepthi et al.[10] In this paper ,Author studied different classification algorithm and concluded that tree-based classifier are best and gives better accuracy for phishing URL detection. Also Author uses various Volume 3, Issue 7, September-October-2018 | http:// ijsrcseit.com Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 398 features such as lexical features, URL based feature, network based

features and domain based feature.

Luong et al. [11] proposed new technique to detect phishing website. In proposed method, Author used six heuristics that are primary domain, sub domain, path domain, page rank, and alexa rank, alexa reputation whose weight and values are evaluated. This approach gives 97 % accuracy but still improvement can be done by enhancing more heuristics.

Ahmad et al.[12] proposed three new features to improve accuracy rate for phishing website detection. In this paper, Author used both type of features as commonly known and new features for classification of phishing and non-phishing site. At the end author has concluded this work can be enhanced by using this novel features with decision tree machine learning classifiers.

2.2 REFERENCES

[1] Routhu Srinivasa Rao1 , Alwyn Roshan Pais :Detection of phishing websites using an efficient feature-based machine learning framework :In Springer 2018. Volume 3, Issue 7, September-october-2018 | http:// ijsrcseit.com Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 399

[2] Chunlin Liu, Bo Lang : Finding effective classifier for malicious URL detection : InACM,2018

[3] Sudhanshu Gautam, Kritika Rani and Bansidhar Joshi : Detecting Phishing Websites Using Rule-Based Classification Algorithm: A Comparison : In Springer,2018.

[4] M. Amaad Ul Haq Tahir, Sohail Asghar, Ayesha Zafar, Saira Gillani : A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms :In International Conference on Computational Science and Computational Intelligence IEEE ,2016.

[5] Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto-Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017.

[6] Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017

[7] Bhagyashree E. Sananse, Tanuja K. Sarode : Phishing URL Detection: A Machine Learning and Web Mining-based Approach : In International Journal of ComputerApplications,2015

[8] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification Phishing Websites Based on URL : IEEE,2015

[9] Priyanka Singh, Yogendra P.S. Maravi, Sanjeev Sharma : Phishing Websites Detection through Supervised Learning Networks : In IEEE,2015

[10] Pradeepthi. K V and Kannan. A: Performance Study of Classification Techniques for Phishing URL Detection: In 2014 Sixth International Conference on Advanced Computing(ICoAC) IEEE,2014

[11] Luong Anh Tuan Nguyen†, Ba Lam To† ,Huu Khuong Nguyen† and Minh Hoang Nguyen : Detecting Phishing Web sites: A Heuristic URL-Based Approach: In The 2013 International Conference on Advanced Technologies for Communications (ATC'13)

[12] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: A Phishing Detection Approach :In IEEE,2013.

[13] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey: An Assessment of Features

Related to Phishing Websites using an Automated Technique:In The 7th International Conference for Internet Technology and Secured Transactions,IEEE,2012

Mohammad et al. [13] proposed model that automatically extracts important features for phishing website detection without requiring any human intervention. Author has concluded in this paper that the process of extracting feature by their tool is much faster and reliable than any manual extraction

2.3 PROBLEM STATEMENT DEFENETION

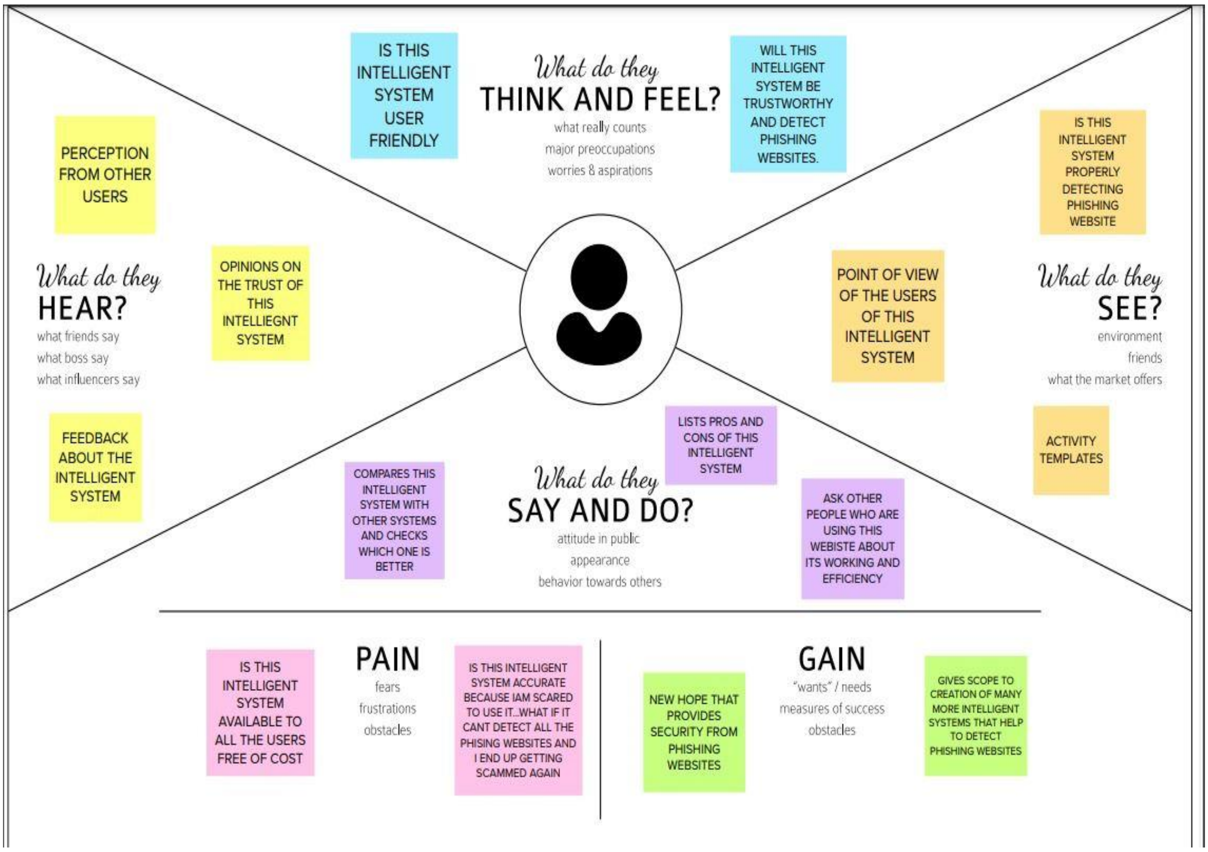
In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking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

Internet has dominated the world by dragging half of the world's population exponentially into the cyber world. With the booming of internet transactions, cybercrimes rapidly increased and with anonymity presented by the internet, Hackers attempt to trap the end-users through various forms such as phishing, SQL injection, malware, man-in-the-middle, domain name system tunnelling, ransomware, web trojan, and so on. Among all these attacks, phishing reports to be the most deceiving attack. Our main aim of this paper is classification of a phishing website with the aid of various machine learning techniques to achieve maximum accuracy and concise model.

CHAPTER 3

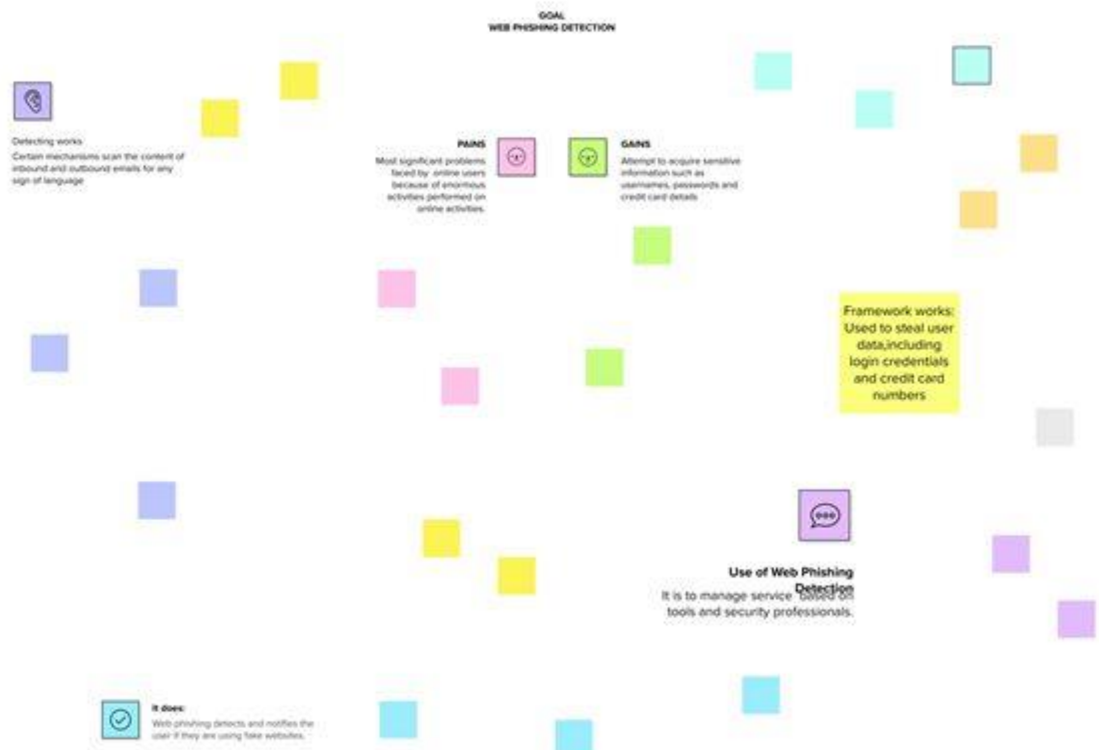
IDEATION & PROPOSED SOLUTION

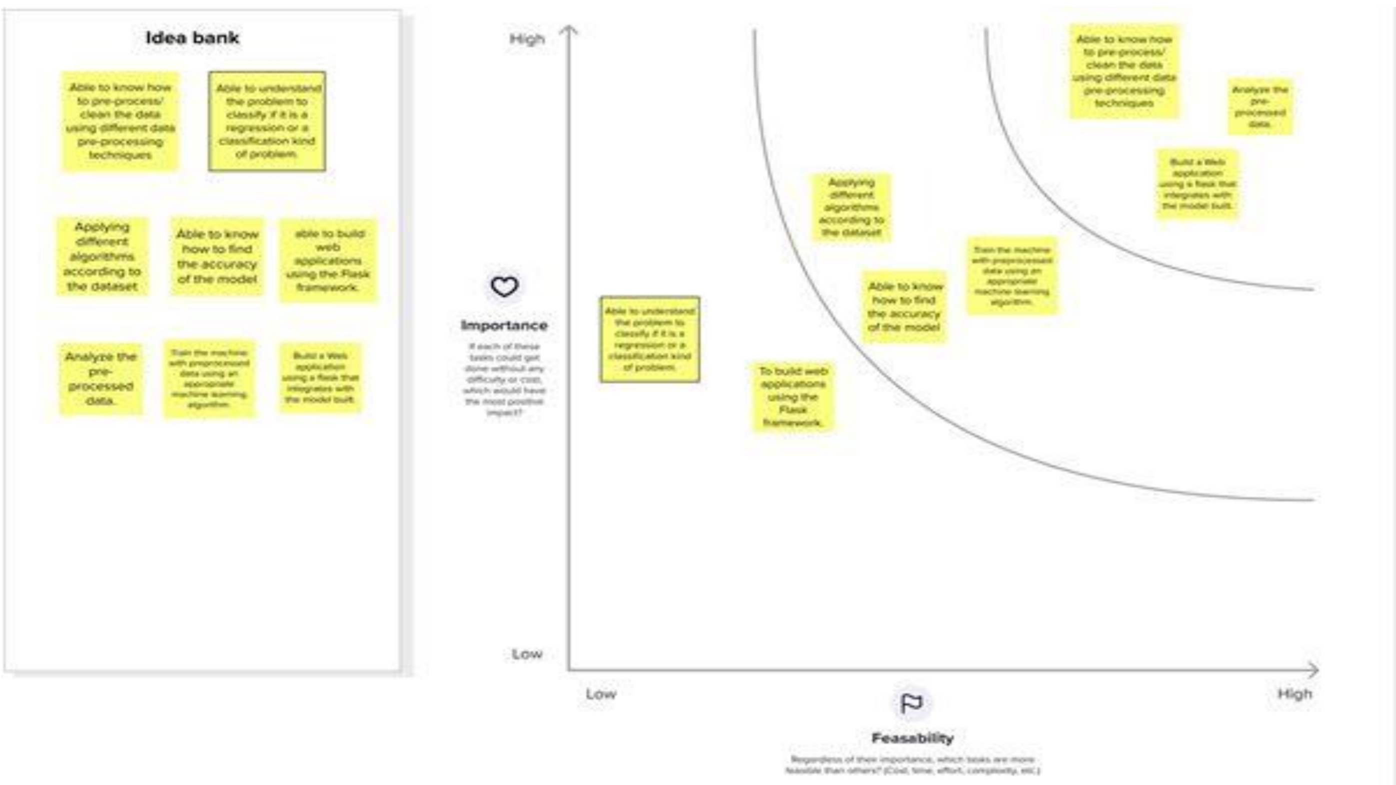
3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Team Gathering, Collaboration and Select the Problem Statement


GOAL
WEB PHISHING DETECTION
Detecting works
Certain mechanisms scan the content of inbound and outbound emails for any sign of language
PAINS
Most significant problems faced by online users because of enormous activities performed on online activities.
GAINS
Attempt to acquire sensitive information such as usernames, passwords and credit card details
Framework works: Used to steal user data,including login credentials and credit card numbers
Use of Web Phishing Detection
It is to manage service based on tools and security professionals.
It does:
Web phishing detects and notifies the user if they are using fake websites.

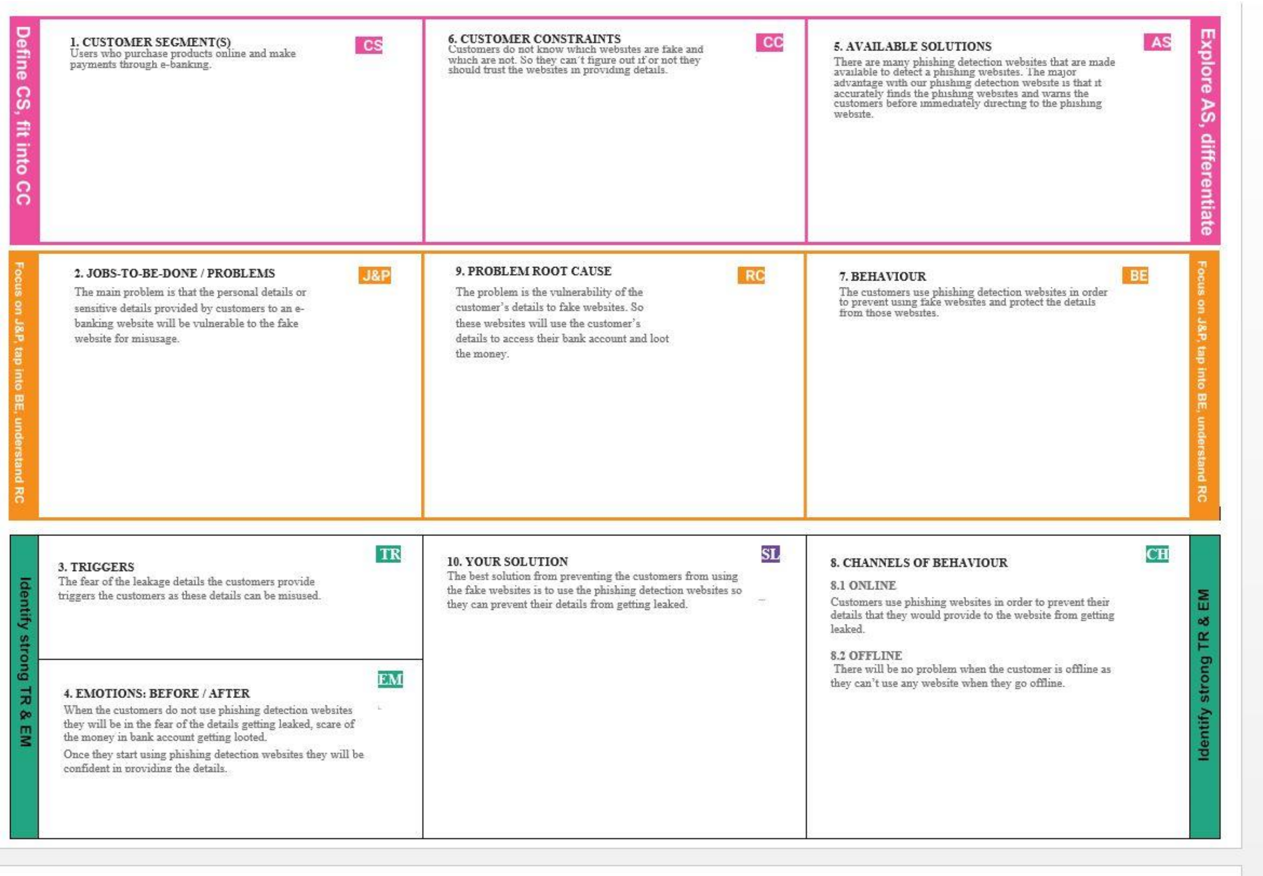

Idea bank
Able to know how to pre-process/ clean the data using different data pre-processing techniques
Able to understand the problem to classify if it is a regression or a classification kind of problem.
Applying different algorithms according to the dataset
Able to know how to find the accuracy of the model
able to build web applications using the Flask framework.
Analyze the pre-processed data.
Train the machine with preprocessed data using an appropriate machine learning algorithm.
Build a Web application using a flask that integrates with the model built.
High
Low
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Able to understand the problem to classify if it is a regression or a classification kind of problem.
To build web applications using the Flask framework.
Applying different algorithms according to the dataset
Able to know how to find the accuracy of the model
Train the machine with preprocessed data using an appropriate machine learning algorithm.
Able to know how to pre-process/ clean the data using different data pre-processing techniques
Analyze the pre-processed data.
Build a Web application using a flask that integrates with the model built.
Low
High
Feasability
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved	• Web phishing aims to steal private information, such as usernames, passwords, and credit card details, byway of impersonating a legitimate entity. • It will lead to information disclosure and property damage. • Large organizations may get trapped indifferent kinds of scams.
2.	Idea / Solution description	In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.
3.	Novelty / Uniqueness	The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once auser makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.
4.	Social Impact / Customer Satisfaction	The feasibility of implementing this idea is moderate neither easy nor tough because the system needs to satisfy the basic requirements of the customer as well as it should act as a bridge towards achieving high accuracy on
		predicting and analysing the detected websites or files to protect our customerto the fullest.

5.	Business Model (Revenue Model)	People buy subscription annually,to protect their files both locally and at remote location with the help of our cloud integrated flask app for web phishing detection.
6.	Scalability of the Solution	By implementing this system, the people can efficiently and effectivelyto gain knowledge about the web phishing techniques and ways to eradicate them by detection . This system can also be integrated with the future technologies

3.4 Problem Solution fit:



REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail Registration by creating a new user name and password
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Login using the credentials we have used during registration
FR-4	User permission	User must give permission access to the search engine so the intelligent system can detect phishing websites
FR-5	Using the intelligent system	User will use the intelligent system to detect phishing websites and save himself from his money being looted

4.2Non-functional Requirements:

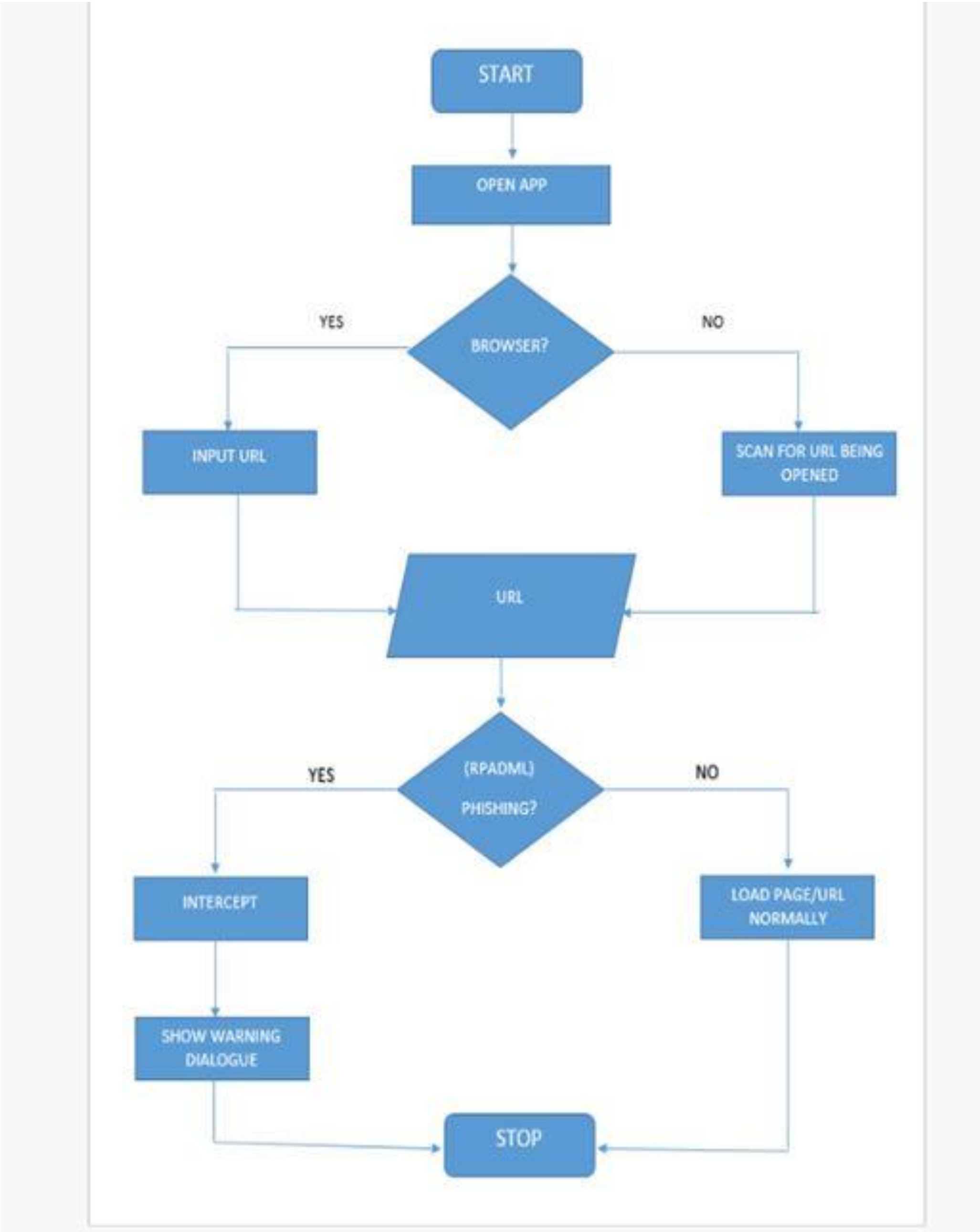
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is very user friendly, any people with less knowledge also can easily understand that they are using the fake website through our alert message.
NFR-2	**Security**	It is very secured as one cannot hack our detection website so one can easily trust our detection website and they will be saved from financial and information loss.
NFR-3	**Reliability**	It has good consistency and performance as it actively detects the fake websites and protect the confidential information and financial loss of the user.
NFR-4	**Performance**	The performance of web phishing detection is high and it is very efficient as it is very easy to understand and has a high security nd scalable
NFR-5	**Availability**	This detection website is available at any system like laptop , mobile phone , desktop and user friendly
NFR-6	**Scalability**	The total execution time of our approach in phishing webpage detection is around 2-3 sec, which is quite less and acceptable environment. As input size increases the execution time increases and this makes the system difficult to handle increases the stress.

PROJECT DESIGN

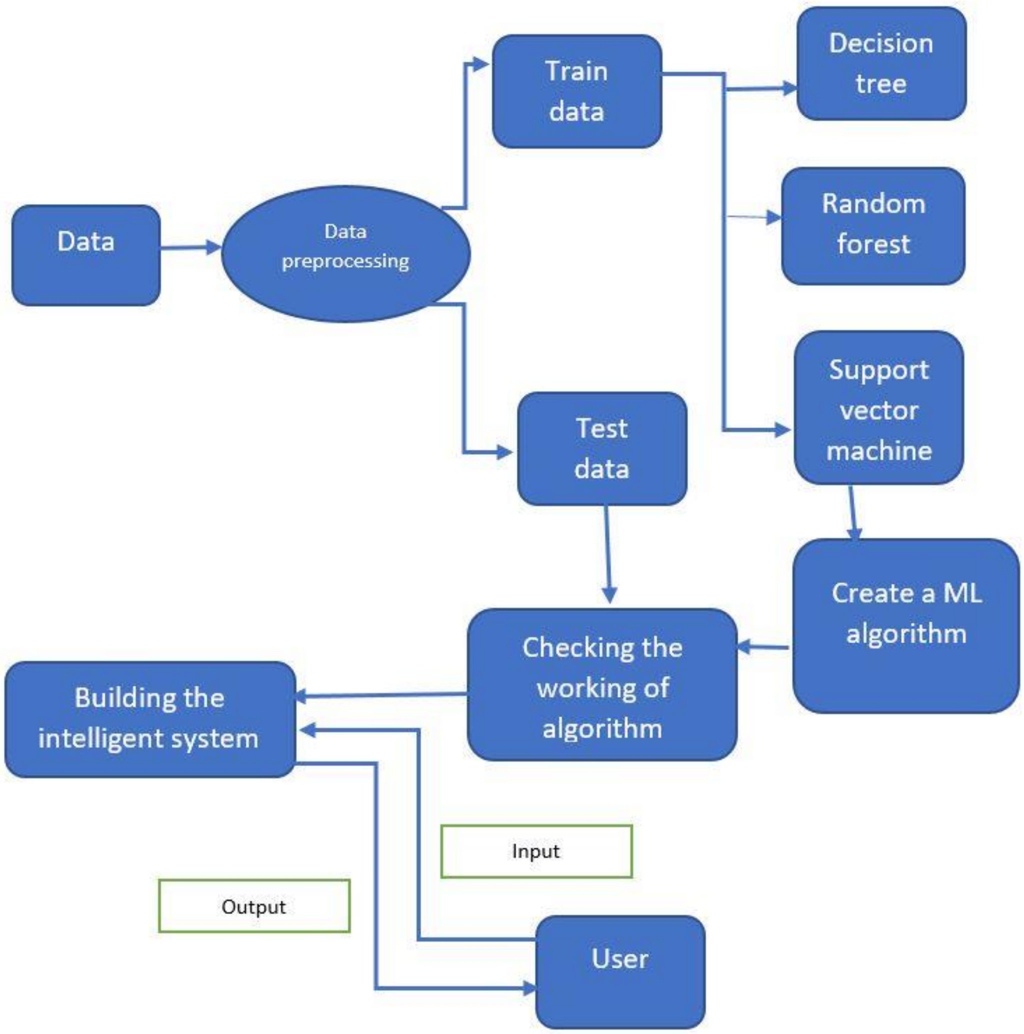
5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



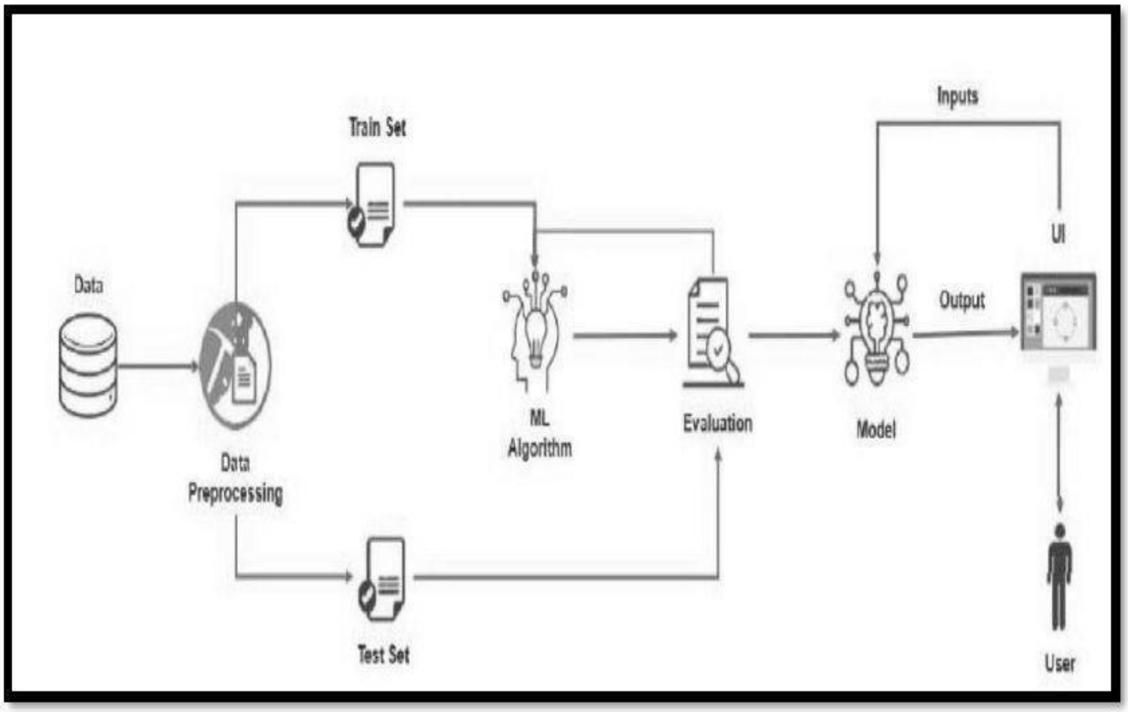
5.2 Solution and Technical Architecture

Solution Architecture



Technical Architecture:

MODEL FOR WEB PHISHING DETECTION


Inputs
Train Set
UI
Data
Output
ML
Algorithm
Evaluation
Model
Data
Preprocessing
Test Set
User

5.3 USER STORIES

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

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard					
Customer (Web user)	User input	USN-1	As a user i can input the particular URL in the required field and waiting for validation.	I can go access the website without any problem	High	Sprint-1
Customer Care Executive	Feature extraction	USN-1	After i compare in case if none found on comparison then we can extract feature using heuristic and visual similarity approach.	As a User i can have comparison between websites for security.	High	Sprint-1
Administrator	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms such as Logistic Regression, KNN	In this i can have correct prediction on the particular algorithms	High	Sprint-1
	Classifier	USN-2	Here i will send all the model output to classifier in order to produce final result.	I this i will find the correct classifier for producing the result	Medium	Sprint-2

PROJECT PLANNING & SCHEDULING

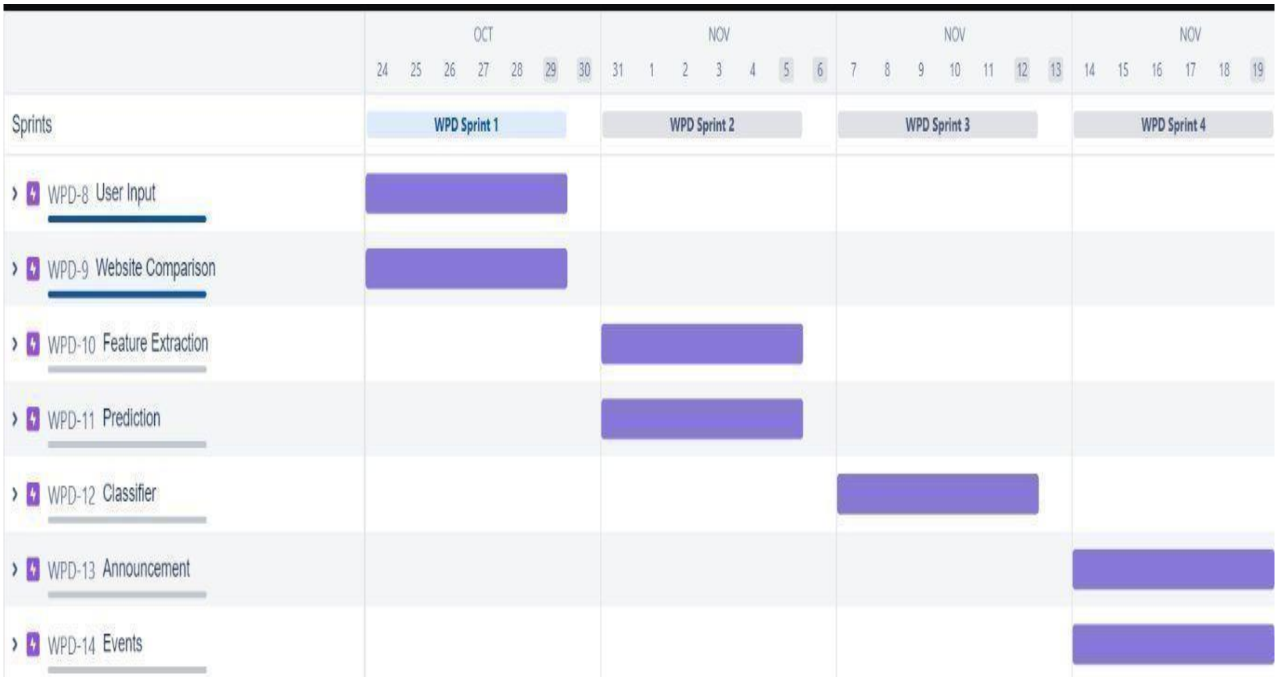
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Feature extraction	USN-4	After comparision, if none found on comparision then extract its feature using heuristic and visual similarity.	10	High	Shalini
Sprint-2	Prediction	USN-5	Model predicts URL using machine learning algorithms such as logistic regression,MLP.	10	Medium	Tejaswi.S
Sprint-2	Accuracy test	USN-6	Selecting best accurate model and process further steps.	15	High	Sruthi.K
Sprint-3	classifier	USN-7	Model sends all output to classifier and produces final result.	5	Medium	Shalini
Sprint-3	Hosting	USN-8	Setting up application and hosting in IBM cloud	10	Medium	Anusha.S
Sprint-4	Announcement	USN-9	Model then displays whether the website is legal site or a phishing site.	15	High	Anusha.S
Sprint-4	Events	USN-10	This model needs the capacity of retrieving and displaying accurate result for a website.	10	High	Shalini

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	7 Days	24 Oct 2022	30 Oct 2022	19	31 Oct 2022
Sprint-2	20	5 Days	31 Oct 2022	04 Nov 2022	18	05 Nov 2022
Sprint-3	20	7 Days	05 Nov 2022	11 Nov 2022	20	11 Nov 2022
Sprint-4	20	8 Days	12 Nov 2022	19 Nov 2022	17	20 Nov 2022

6.3 Reports from JIRA



CHAPTER-7

CODING & SOLUTION

7.1 Feature 1

```
#app.py
# importing required libraries

from feature import FeatureExtraction
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')

file = open("model.pkl", "rb")
gbc = pickle.load(file)
file.close()


app = Flask(_name_)


@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":

        url = request.form["url"]
```

```
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1, 30)

        y_pred = gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0, 0]
        y_pro_non_phishing = gbc.predict_proba(x)[0, 1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template('index.html', xx=round(y_pro_non_phishing, 2), url=url)
    return render_template("index.html", xx=-1)


if __name___== "_main_":
    app.run(debug=True, port=2002)
```

7.2 Feature 2

```
#feature.py
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\.g
d|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\.ly', url)
            ip_address = socket.gethostbyname(self.domain)
            ip_match =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.21
1\.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|10
7\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|'
'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\
.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|'
'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213
\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|'
'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\
.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|'
'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.1
56\.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
```

CHAPTER 8
TESTING

8.1 Test Cases

Date	15-Nov-22
Team ID	PNT2022TMID15402
Project Name	Project - Web Phishing Detection
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		Shalini A
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		K.Siva Sruthi
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		S.Tejaswi
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		S.Anusha
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://github.com/tejaswireddy9 2. https://rmd.ac.in/ 3. https://rmd279.examly.io/login 4. https://www.annauniv.edu/nwsnew/5. https://www.google.com/ 6. https://cloud.ibm.com/login?redirect=%2Faccount%2Fsettings	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		S.Tejaswi

8.2 User Acceptance Testing

UAT Execution & Report Submission

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Web Phishing Detection] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36

Not Reproduced	0	0	1	0	1

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

3. Test Case Analysis

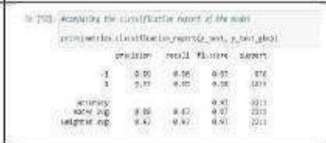
This report shows the number of test cases that have passed, failed, and untested

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

CHAPTER 9

RESULTS

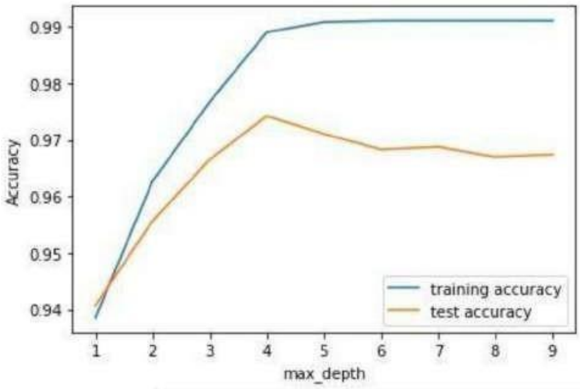
9.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** **Gradient Boosting Classification** Accuray Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

1. METRICS:
CLASSIFICATION REPORT:

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

         print(metrics.classification_report(y_test, y_test_gbc))

              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

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

PERFORMANCE :



Out[83]:

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

2. TUNE THE MODEL – HYPERPARAMETER TUNING

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

Out[58]:
GridSearchCV

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

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

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

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

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

VALIDATION METHODS: KFOLD & Cross Folding

Wilcoxon signed-rank test

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

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

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

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

Out[78]: 95.0
```

5x2CV combined F test

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

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

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

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

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

CHAPTER -10

Advantages of web phishing detection

1. Improve on Inefficiencies of SEG and Phishing Awareness Training
2. It Takes a Load off the Security Team
3. It Offers a Solution, Not a Tool
4. Separate You from Your Competitors
5. This system can be used by many e-commerce websites in order to have good customer relationships.
6. If internet connection fails this system will work

Disadvantages of web phishing detection

1. All website related data will be stored in one place.
2. It is a very time-consuming process.

CHAPTER 11

CONCLUSION

It is outstanding that a decent enemy of phishing apparatus ought to anticipate the phishing assaults in a decent timescale. We accept that the accessibility of a decent enemy of phishing device at a decent time scale is additionally imperative to build the extent of anticipating phishing sites. This apparatus ought to be improved continually through consistent retraining. As a matter of fact, the accessibility of crisp and cutting-edge preparing dataset which may gained utilizing our very own device [30, 32] will help us to retrain our model consistently and handle any adjustments in the highlights, which are influential in deciding the site class. Albeit neural system demonstrates its capacity to tacklea wide assortment of classification issues, the procedure of finding the ideal structure is verydifficult, 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.

CHAPTER-12

Future Scope

There is a scope for future development of this project. We will implement this using advanced deep learning method to improve the accuracy and precision. Enhancements canbe done in an efficient manner. Thus, the project is flexible and can be enhanced at any time with more advanced features.

CHAPTER-13

Appendix:

1. Application Building
2. Collection of Dataset
3. Data Pre-processing
4. Integration of Flask App with IBM Cloud
5. Model Building
6. Performance Testing
7. Training the model on IBM
8. User Acceptance Testing
9. Ideation Phase
10. Preparation Phase
11. Project Planning
12. Performance Testing
13. User Acceptance Testing

Project Link: https://github.com/IBM-EPBL/IBM-Project-31113-1660196270

Project Demo Link: https://github.com/IBM-EPBL/IBM-Project-31113-1660196270/tree/main/Final%20deliverables/Demo%20video