IBM NALAIYA THIRAN

PROJECT REPORT ON

WEB PHISHING DETECTION

TEAM ID: PNT2022TMID03926

UNIVERSITY COLLEGE OF ENGINEERING (BIT CAMPUS)

Team Members

JANANI U

DHIVYA E

KAYALVIZHI B

VENGATESAN D

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 intheir 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 commonsocial engineering method that mimics trustful uniform resource locators (URLs) and webpages. The objective of this project is to train machine learning models anddeep 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 pre ferred 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 real time URLs and produce effective results.
- To implement the concept of RNN, which is a familiar ML technique 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

SURVEY ON PAPERS RELATED TO WEB PHISING DETECTION

NAME: Detection of E-mail phising attacks using Machine Learning
AUTHOR:DhruvRathee, Sumanmann.
PUBLISHED YEAR: 2022

BASIC DESCRIPTION:

Phishing is the most prominent cyber-crime that uses camouflaged e-mail as a weapon. It is defined as the strategy adopted by fraudsters in-order-to get private details from persons by professing to be from well- known channels like offices, bank, or a government organization. In this era of modernization, electronic mails are accustomed globally as communiqué channel for both private and professional purposes. The particulars exchanged over e- mails are often confidential and sensitive for example info of bank statements, payment bills, debit-credit reports, and authentication data. This makes e-mails precious for hackers because they can exploit these details for maleficent intends. The main goal of the attackers is to acquire personal details by deceiving the e-mail recipient to click noxious link or download the attachment under false pretences.

HIGHLIGHTS OF THE PROJECT:

From this project report we understood that the following results,

1. Planning—this involves identifying the targets, the information sought, andcreating/identifying the tools and techniques that will be used in the attack (such as emails with malicious links and the spoof sites these links direct to).
2. Phishing—the stage during whichthe identified targets are phished using the resources created in Stage 1.
3. Infiltration—depending on the method used, this stage will vary but it essentially consists of the response from the target and gaining access to the personal information sought.
4. Data collection and exploitation—this is the stage at which the phisher extracts the information sought and utilizes it to achieve the ends established during the planning phase. This often involves fraud whereby the attackers impersonate the victims to access their accounts, etc. Another common occurrence is the selling of this personal data on the online black market.
5. Exfiltration—finally, the phisher attempts to remove as much evidence of their attempt as possible (such as the deletion of fake sites). There may also be some analysis on the success of the attack and assessment of future attacks.

OVER VIEW OF THE PROJECT:

The person who wants to escape from the fake websites will be informed about the phishing websites to the user immediately .Phishing e-mails have transformed and modified into numerous categories, some are well-crafted to appear legitimate**.** The phishing e-mails are sent from authorized accounts, so that the mails seems to appear legit. To accomplish their mission, attackers often keep track of the user personal details. They always send mails from only those accounts that belong to a friend or a previous business colleague by tricking the victim with fake e-mail IDs. The criminals often work very hard to make e-mails appear genuine by entailing believable-wordings, graphic-interface, and logos. Although ,phished e-mails do not appear similar as actual phishing emails are custom-built for their anticipated objectives.

Machine learning-based approaches help in detecting phishing attacks more efficiently by giving lower false-positive rates and high accuracy in comparison with other methods.

Nine features were extracted from the e-mail and the last few features were obtained from the WHOIS query. They used larger datasets of about 7000 normal e-mails and 860 phishing e-mails for training and testing the classifier. They focused specifically on URL properties which might not be the appropriate technique because identification of phished emails depends on various factors. Also, criminals could use tools to obfuscate URLs such as tiny URLs (https://tiny.qe/) and design them to appear legal. Their filter scores 97.6% measure, false-positive rate of 0.13%, and a false-negative rate of 3.6%.

RESULT:

From this survey a number of anti-phishing softwaretechniques. Some of the important aspects in measuring phish-ing solutions are:

• Detection accuracy with regards to zero-hour phishingattacks. This is due to the fact that phishing websites aremostly short-lived and detection at hour zero is critical.

• Low false positives. A system with high false positivesmight cause more harm than good. Moreover, end-userswill get into the habit of ignoring security warnings if the
classifier is often mistaken.

Generally, software detection solutions are:

• Blacklists.
• Rule-based heuristics.
• Visual similarity.
• Machine Learning-based classifiers.

CONCLUSION:

The paper presents a survey analysis of actual phishing email identification works from various perspectives. This survey is unique in the sense that it relates works to their openly available tools and resources.Many ML methods have been adopted to identify phishing emails, but these cannot effectively detect new phishing scams, which needs significant manual feature engineering.Anti-phishing technology developed on the source code features is quite slow in terms of the classification of phishing emails given its dependence on third-

STATEMENT DEFENETION

In order to detect party services and scraping of the email content. The analysis of the presented works revealed that not much work had been performed on phishing email detection using natural level Natural Language Processing (NLP) techniques. Therefore, many open issues are associated with this phishing email detection.

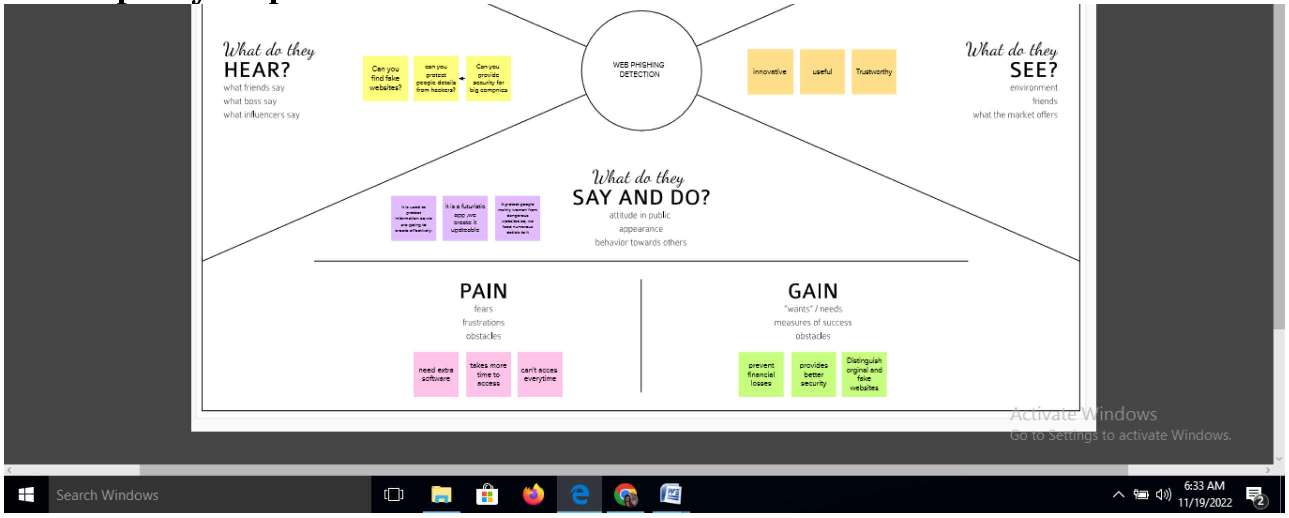
2.3 PROBLEM and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking 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 tunneling, 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.

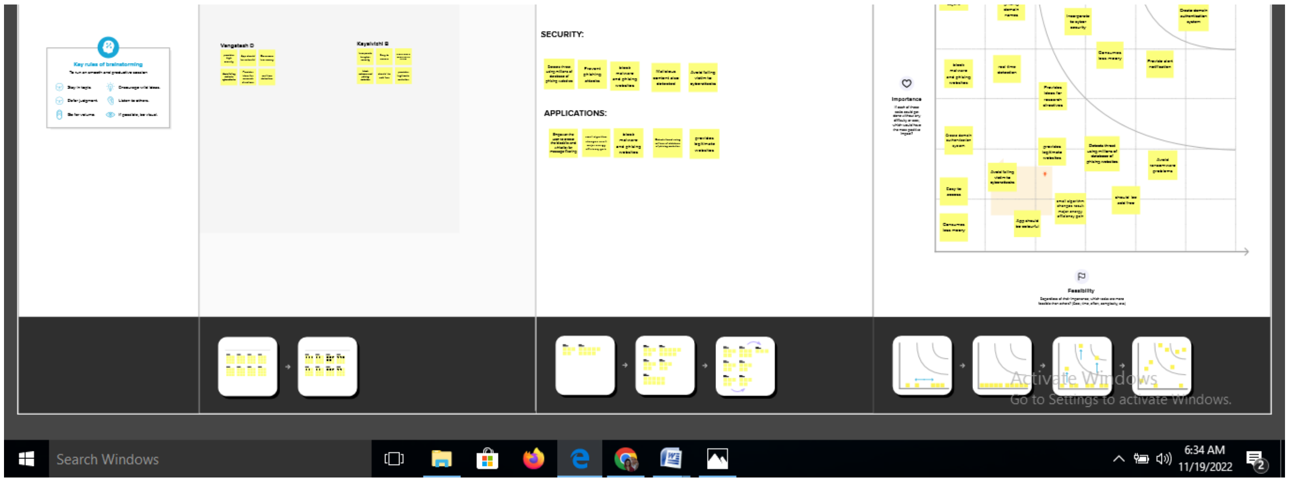
CHAPTER 3

<u>IDEATION & PROPOSED SOLUTION</u>

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

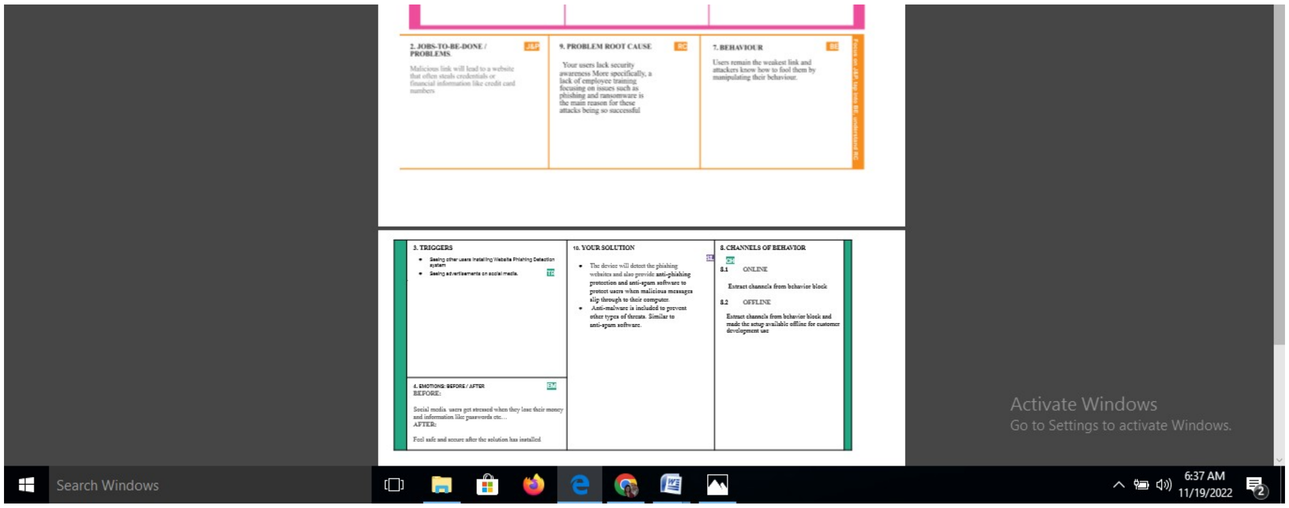

Key rules of brainstorming
Vengatesh D
Kayalvizhi B
SECURITY:
Prevent phishing attacks
block malware and phishing websites
Malicious content also detected
Avoid falling victim to cyberattacks
APPLICATIONS:
block malware and phishing websites
provides legitimate websites
Importance
Feasibility
Activate Windows
Go to Settings to activate Windows.
Search Windows
6:34 AM
11/19/2022

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	Our aim is to helps the user by detecting phishing websites in network platform.
2.	Idea / Solution description	Provides best security against phishing websites to the users
3.	Novelty / Uniqueness	Proposed solution provides information to identify and block phishing websites using a variety of methods.
4.	Social Impact / Customer Satisfaction	Malicious links will lead to a website that often steals login credentials or financial information like credit card numbers. This will protect users from phishing websites and protects their personal information.

Business Model (Revenue Model)	Our model targets the network users who don’t want to lose their valuable personal information to the phishing websites. It gains profit in the market as the targeted users are all networks users.
Scalability of the Solution	The design works with same efficiency regardless of the user’s device. The performance of our design will not depends on the traffic of the users.

3.4 Problem Solution fit:



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

Malicious link will lead to a website that often steals credentials or financial information like credit card numbers

9. PROBLEM ROOT CAUSE RC

Your users lack security awareness More specifically, a lack of employee training focusing on issues such as phishing and ransomware is the main reason for these attacks being so successful

7. BEHAVIOUR BE

Users remain the weakest link and attackers know how to fool them by manipulating their behaviour.

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

3. TRIGGERS TR

- Seeing other users Installing Website Phishing Detection system
- Seeing advertisements on social media.

4. EMOTIONS: BEFORE / AFTER EM

BEFORE:

Social media users get stressed when they lose their money and information like passwords etc…

AFTER:

Feel safe and secure after the solution has installed

10. YOUR SOLUTION SL

- The device will detect the phishing websites and also provide **anti-phishing protection and anti-spam software to protect users when malicious messages slip through to their computer.**
- **Anti-malware is included to prevent other types of threats. Similar to anti-spam software.**

8. CHANNELS OF BEHAVIOR CH

8.1 ONLINE

Extract channels from behavior block

8.2 OFFLINE

Extract channels from behavior block and made the setup available offline for customer development use

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

	FR No. Functional Requirement (Epic)	**Sub Requirement (Story / Sub-Task)**
FR-1	Verifying input	User inputs an URL (Uniform Resource Locator) innecessary field to check its validation.
FR-2	Website Evaluation	Model evaluates the website using Blacklist and Whitelist approach
FR-3	Extraction and Prediction	It retrieves features based on heuristics and visual similarities. The URL is predicted by the model using Machine Learning methods such as Logistic Regressionand KNN.
FR-4	Real Time monitoring	The use of Extension plugin should provide a warningpop-up when they visit a website that is phished. Extension plugin will have the capability to also detectlatest and new phishing websites
FR-5	Authentication	Authentication assures secure site, secure processes and enterprise information security.

4.2 Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Analysis of consumers' product usability in the design process with user experience as the core maycertainly help designers better grasp users' prospective demands in web phishing detection, behaviour, and experience.
NFR-2	**Security**	It guarantees that any data included within the system or its components will be safe from malwarethreats or unauthorised access. If you wish to prevent unauthorised access to the admin panel, describe the login flow and different user roles as system behaviour or user actions.
NFR-3	**Reliability**	It specifies the likelihood that the system or itscomponent will operate without failure for a specified amount of time under prescribed conditions.
NFR-4	**Performance**	It is concerned with a measurement of the system's reaction time under various load circumstances.
NFR-5	**Availability**	It represents the likelihood that a user will be able to access the system at a certain moment in time. While it can be represented as an expected proportion of successful requests, it can also be defined as a percentage of time the system is operational within a certain time period.
NFR-6	**Scalability**	It has access to the highest workloads that will allow the system to satisfy the performance criteria. There are two techniques to enable the system to grow as Vertical and horizontal scaling.

CHAPTER 5

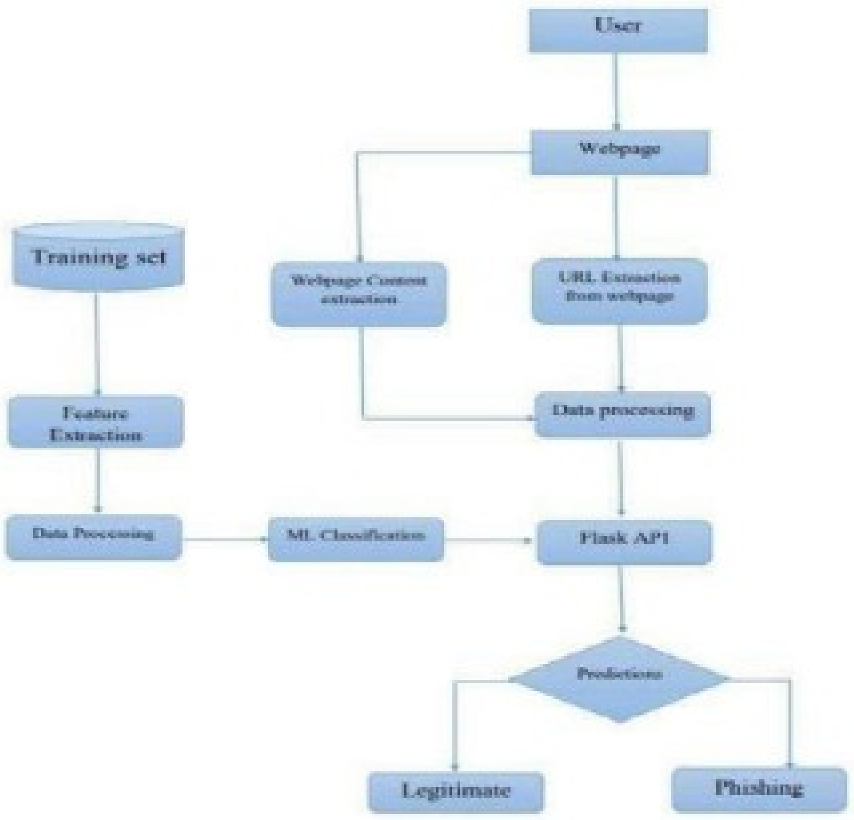
PROJECT DESIGN

5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a

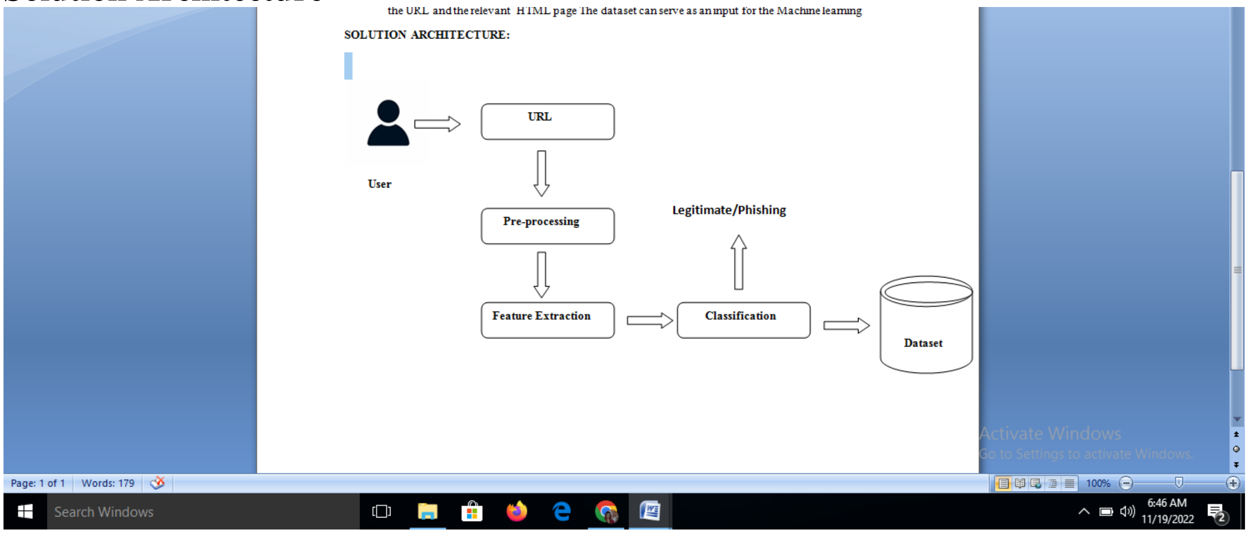
system. A neat and clear DFD can depict the right amount of the system requirement graphically. It

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



5.2 Solution and Technical Architecture

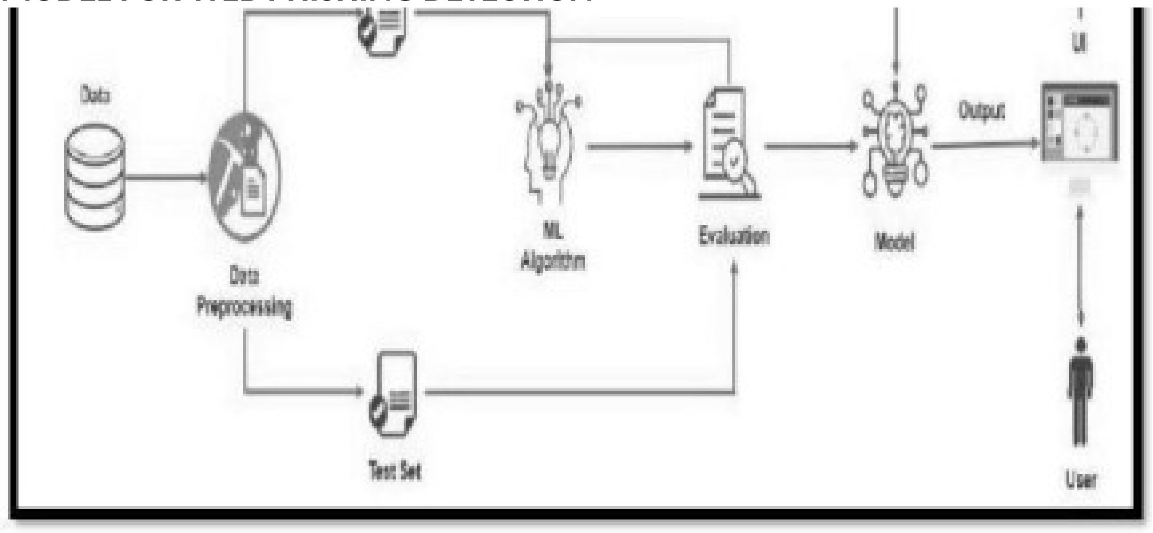
Solution Architecture


the URL and the relevant HTML page The dataset can serve as an input for the Machine learning

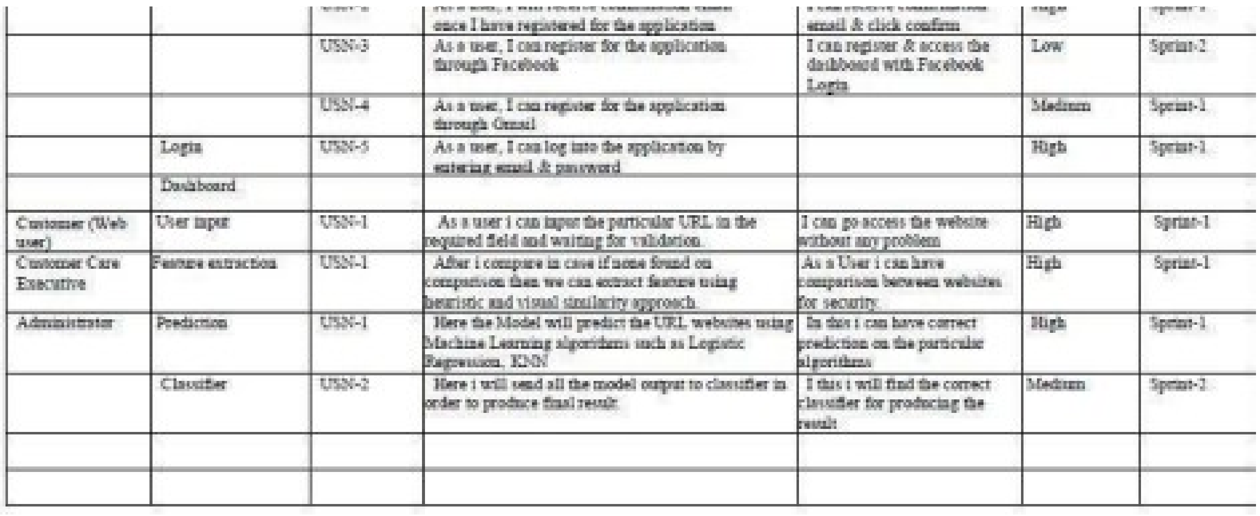
SOLUTION ARCHITECTURE:

Technical Architecture:

MODEL FOR WEB PHISHING DETECTION



5.3 USER STORIES

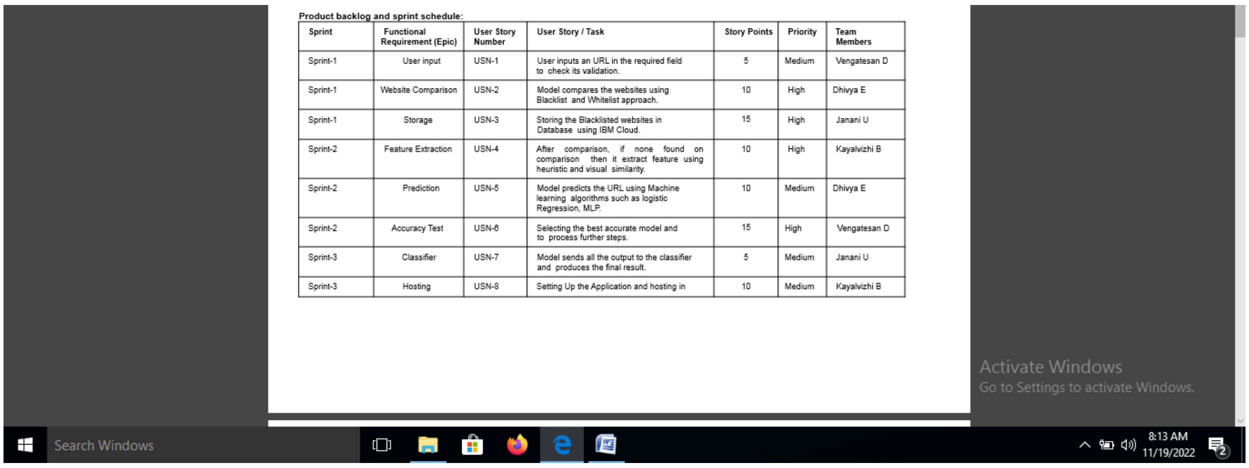


		[illegible]	[illegible] once I have registered for the application	[illegible] email & click confirm	[illegible]	[illegible]
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register 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

CHAPTER 6

PROJECT PLANNING & SCHEDULING

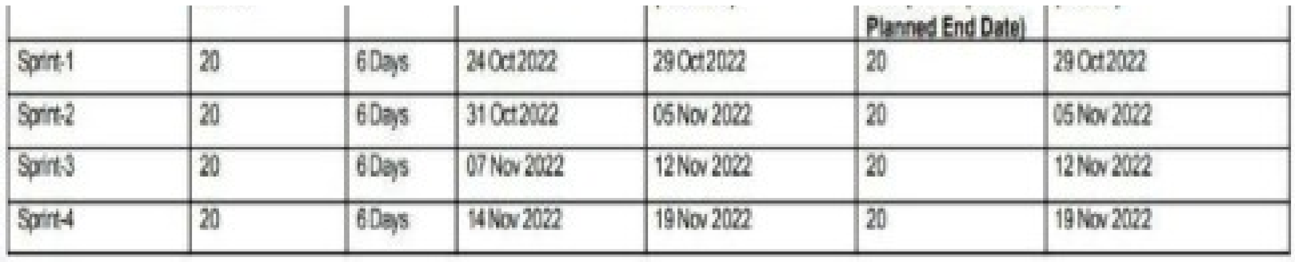
6.1 Sprint Planning & Estimation



Product backlog and sprint schedule:

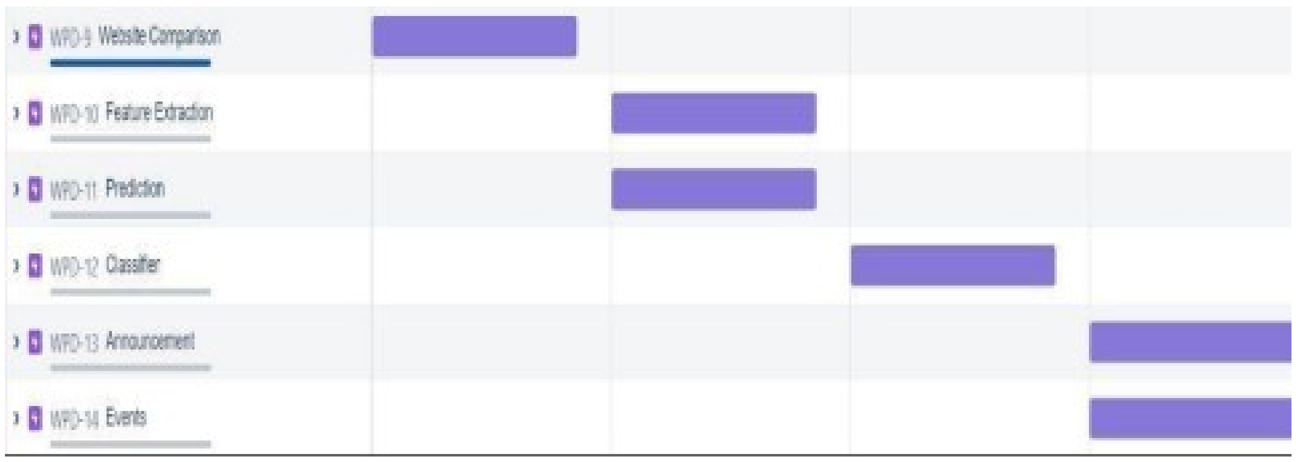
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	User inputs an URL in the required field to check its validation.	5	Medium	Vengatesan D
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklist and Whitelist approach.	10	High	Dhivya E
Sprint-1	Storage	USN-3	Storing the Blacklisted websites in Database using IBM Cloud.	15	High	Janani U
Sprint-2	Feature Extraction	USN-4	After comparison, if none found on comparison then it extract feature using heuristic and visual similarity.	10	High	Kayalvizhi B
Sprint-2	Prediction	USN-5	Model predicts the URL using Machine learning algorithms such as logistic Regression, MLP.	10	Medium	Dhivya E
Sprint-2	Accuracy Test	USN-6	Selecting the best accurate model and to process further steps.	15	High	Vengatesan D
Sprint-3	Classifier	USN-7	Model sends all the output to the classifier and produces the final result.	5	Medium	Janani U
Sprint-3	Hosting	USN-8	Setting Up the Application and hosting in	10	Medium	Kayalvizhi B

6.2 Sprint Delivery Schedule



					Planned End Date)	
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



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 = "Itis {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

importre

import urllib.request

from bs4 import BeautifulSoup

importsocket

importrequests

from googlesearch importsearch

import whois

from datetime import date, datetime

importtime

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.app
        end(self.shortUrl
        ())
        self.features.app
        end(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(se
lf.DomainRegLen())
self.features.append(se
lf.Favicon())


self.features.append(self.N
onStdPort())
self.features.append(self.H
TTPSDomainURL())
self.features.append(self.R
equestURL())
self.features.append(self.A
nchorURL())
self.features.append(self.Li
nksInScriptTags())
self.features.append(self.S
erverFormHandler())
self.features.append(self.In
foEmail())
self.features.append(self.A
bnormalURL())
self.features.append(self.W
ebsiteForwarding())
self.features.append(self.St
    atusBarCust())


    self.features.append(self.Di
    sableRightClick())
    self.features.append(self.Us
    ingPopupWindow())
    self.features.append(self.Ifr
    ameRedirection())
    self.features.append(self.Ag
    eofDomain())
    self.features.append(self.D
    NSRecording())
    self.features.append(self.W
    ebsiteTraffic())
    self.features.append(self.Pa
    geRank())
    self.features.append(self.G
    oogleIndex())
    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
        defshortUrl(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@
        defsymbol(self):
            ifre.findall("@",self.url):
                return -1
```

```
    return 1


# 5.Redirecting//
def redirecting(self):
    ifself.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
defHppts(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_dat
        e 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(":")
        iflen(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'])]
    ifself.url in img['src'] or self.domain in
        img['src'] orlen(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'])]
    ifself.url in audio['src'] orself.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'])]
    ifself.url in embed['src'] orself.domain in
        embed['src'] orlen(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'])]
    ifself.url in iframe['src'] orself.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'])]

         ifself.url in link['href'] orself.domain in

            link['href'] orlen(dots) == 1: success = success

            + 1

         i = i+1


      for scriptin self.soup.find_all('script',src=True):

         dots = [x.start(0) for x in re.finditer('\.',script['src'])]

         ifself.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

               elifself.url notin form['action'] and self.domain

                   notin form['action']: return 0

               else:

                   return 1

   except:

       return -1


# 17. InfoEmail

 def InfoEmail(self):

   try:

       ifre.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_d
       ate 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_d
           ate 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
            elifip_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	PNT2022TMID32416
Project Name	Project-WebPhishingDetection
Maximum Marks	4marks

Test case ID	Feature Type	Component	TestScenario	Pre-Requisite	Steps ToExecute	TestData	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://phishingshield.herokuapp.com/	ShouldDisplay the Webpage	Workingas expected	Pass		N		**Dhivya E**
LoginPage_TC_OO 2	UI	Home Page	Verify the UI eleme nts is Responsive		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check whether the button is responsive ornot 4. Reload and TestSimultaneously	https://phishingshield.herokuapp.com/	Should Wait for Response and then getsAcknowledge	Workingas expected	Pass		N		Janani U
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4.Observe the results	https://phishingshield.herokuapp.com/	User should observe whether the websiteislegitimate ornot.	Workingas expected	Pass		N		Kayalvizhi B
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website ornot		1. Enter URL and click go 2. Type orcopypaste the URL 3. Check the website is legitimate or not 4. Continue if the website is legitimate orbe cautiousif itis notlegitimate.	https://phishingshield.herokuapp.com/	Application shouldshow that Safe Webpage orUnsafe.	Workingas expected	Pass		N		Vengatesan D
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 orbe cautiousif itis not secure	1. https://avbalajee.github.io/welcome 2.totalpad.com 3. https://www.klnce.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	Workingas expected	Pass		N		Janani U

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**

10	4	2	3
1	0	3	0
2	3	0	1
10	2	4	20
0	0	1	0

By Design 20 Duplicate 4 External 6 Fixed 36 Not Reproduced 1

0	0	0	0
0	0	2	1

Skipped 0 Won't Fix 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**

10	0	0
50	0	0
5	0	0
3	0	0
10	0	0
10	0	0

Print Engine 10 Client Application 50 Security 4 Outsource Shipping 3 Exception Reporting 9 Final Report Output 10 Version Control 4 0 0 4

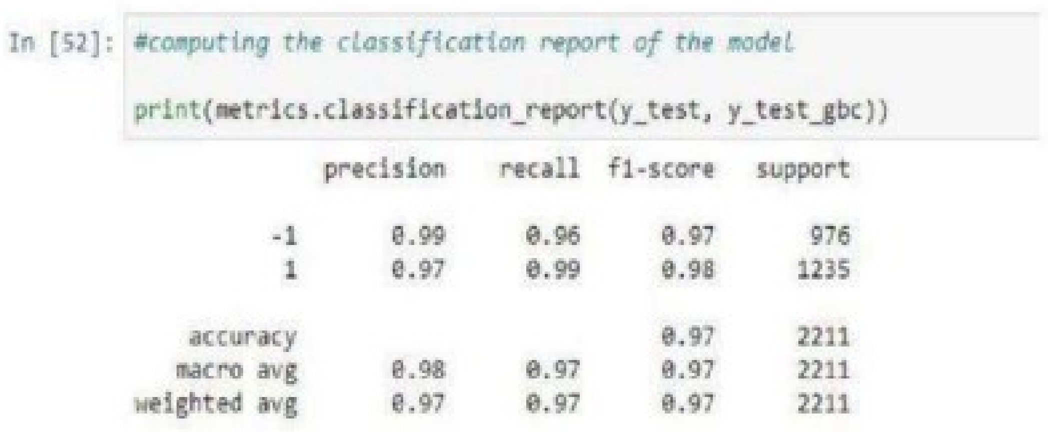
CHAPTER 9

RESULTS

9.1 Performance Metrics

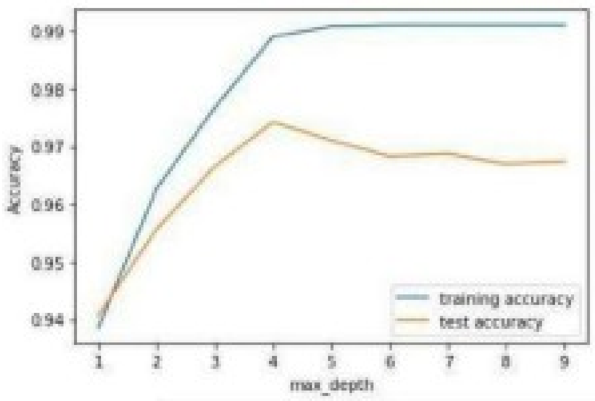
S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** **Gradient Boosting Classification** Accuray Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

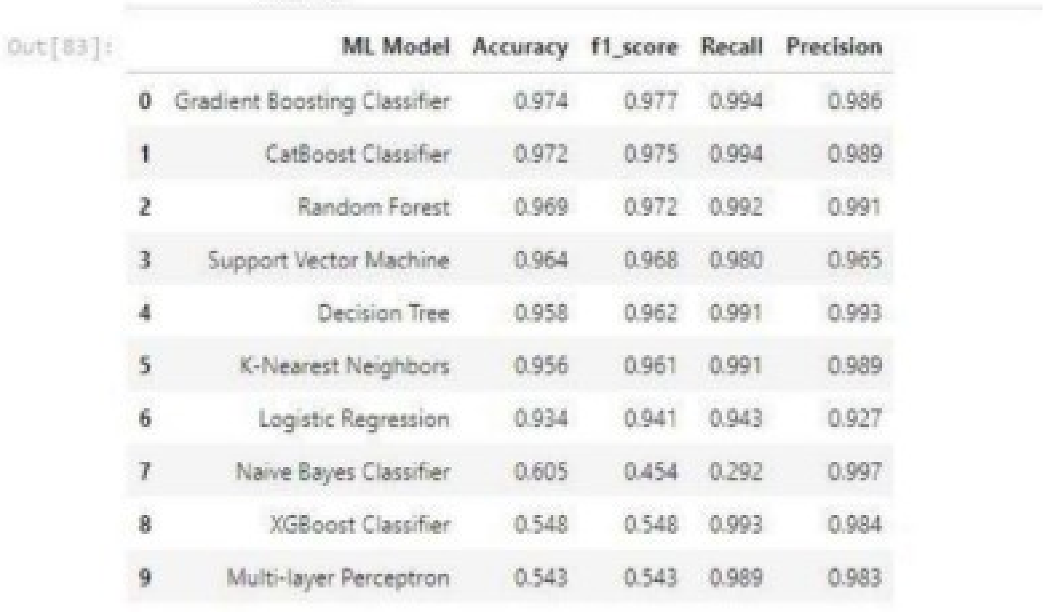
1. METRICS:
CLASSIFICATION REPORT:



```
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
```

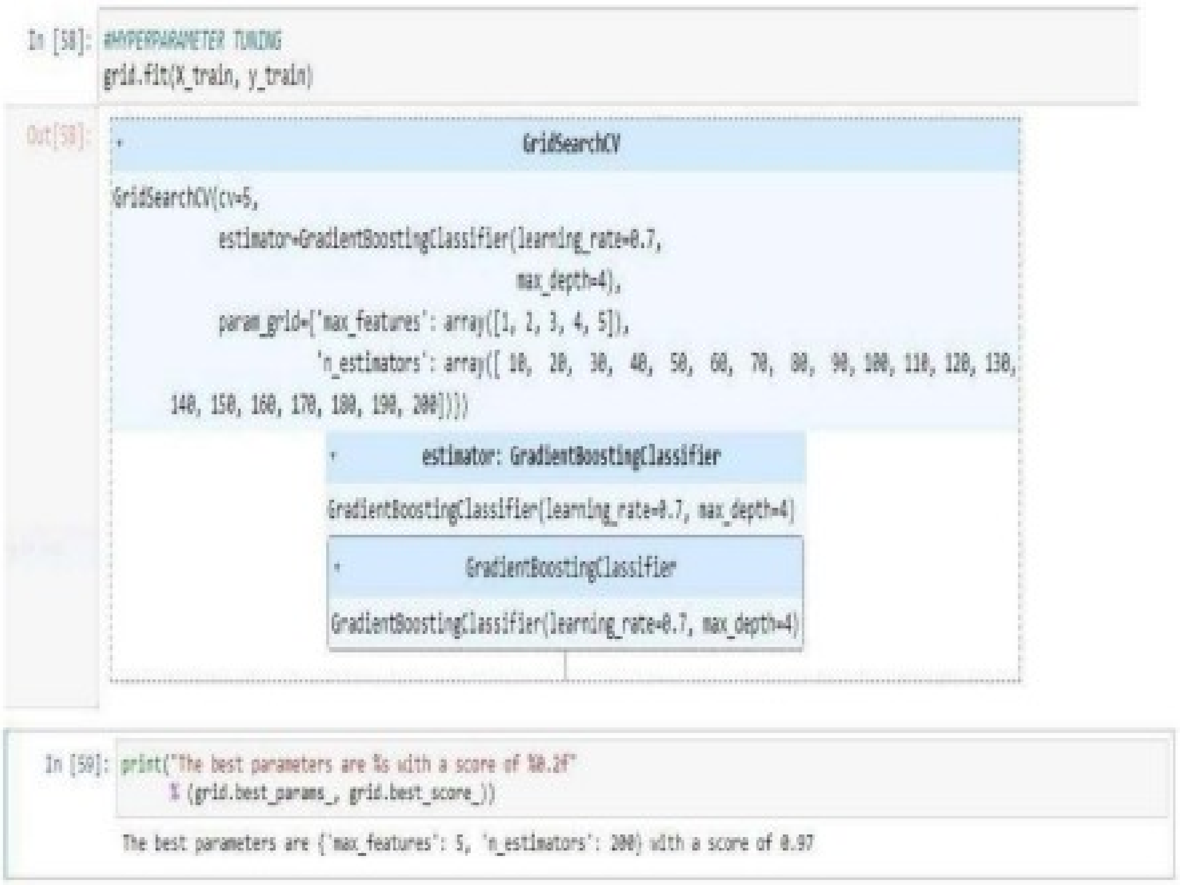
PERFORMANCE :




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



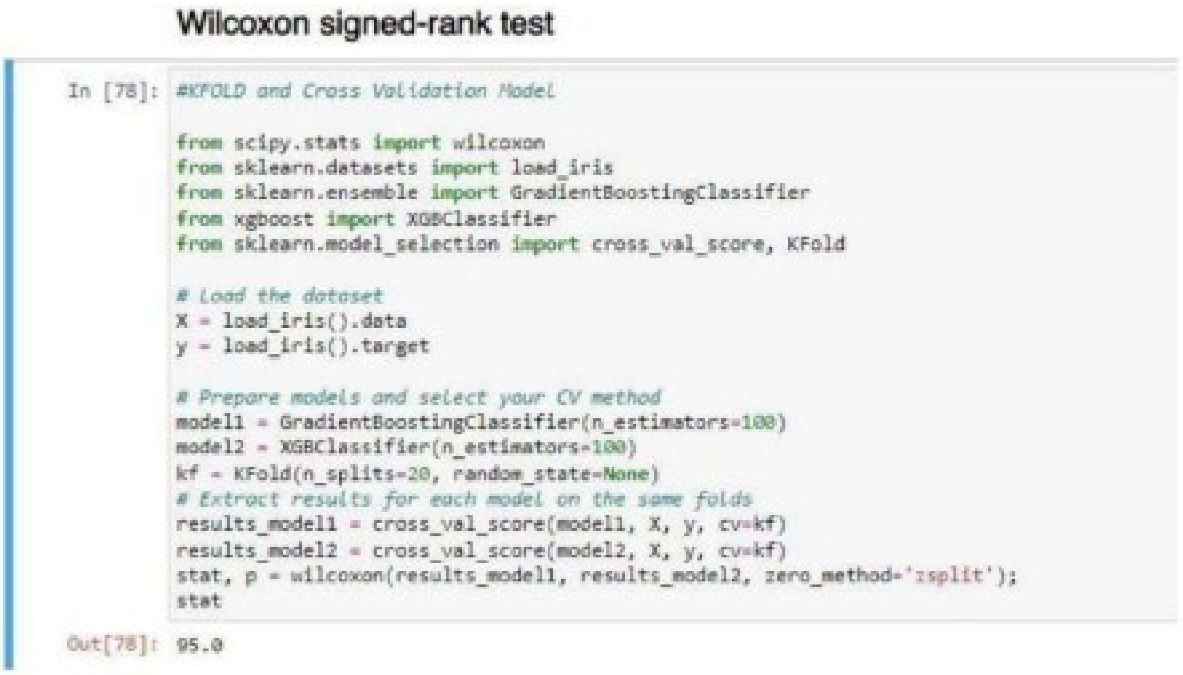
```
In [58]: #HYPERPARAMETER TUNING
         grid.fit(X_train, y_train)
```

```
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
```

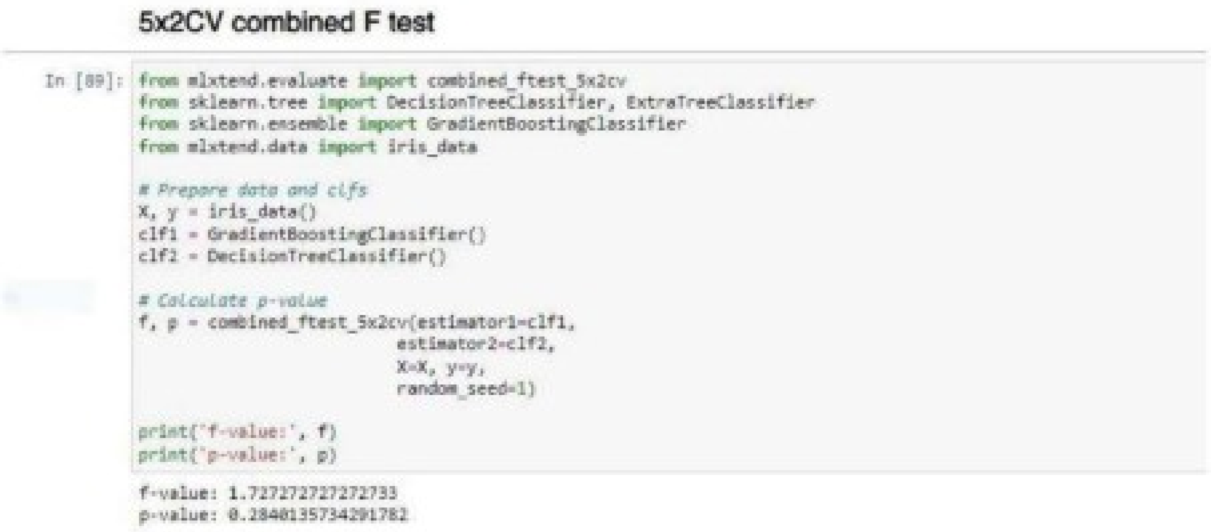
L

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: 1.7272727272727233
         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. ModelBuilding
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-49729-1660837038