WEB PHISHING DETECTION USING MACHINE LEARNING ALGORITHMS

PROJECT REPORT

Submitted by

Aadhish S	193002001
Dineshkumar G	193002024
Krishi Divya Dharshini V	193002050
Manjunathan R	193002057

EEC ELECTIVE

Department of Electronics and Communication Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

ODD SEM 2022 – 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS	ABBREVIATION
DFD	Data Flow Diagram
SVM	Support Vector Machine
DBN	Deep Belief Networks
ISP	Internet Service Provider
UI	User Interface

1. INTRODUCTION

1.1. Project Overview:

Web Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending as a reputable entity or person in email or other communication channels. Typically, a victim receives a message that appears to have been sent by a known contact or organization. The message contains malicious software targeting the user's computer or has links to direct victims to malicious websites to trick them into divulging personal and financial information, such as passwords, account IDs or credit card details. Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer's defence systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization's logos and other legitimate contents.

1.2. Purpose:

Website Phishing costs internet users billions of dollars per year. Phishers steal personal information and financial account details such as usernames and passwords, leaving users vulnerable in the online space. The COVID-19 pandemic has boosted the use of technology in every sector, resulting in shifting of activities like organising official meetings, attending classes, shopping, payments, etc. from physical to online space. This means more opportunities for phishers to carry out attacks impacting the victim financially, psychologically & professionally. In 2013, 450 thousand phishing attacks led to financial losses of more than 5.9 billion dollars. As per CheckPoint Research Security Report 2018, 77% of IT professionals feel their security teams are unprepared for today's cybersecurity challenge. The same report indicates that 64% of organizations have experienced a phishing attack in the past year. Detecting phishing websites is not easy because of the use of URL obfuscation to shorten the URL, link redirections and manipulating links in such a way that it looks trustable and the list goes on. This necessitated the need to switch from traditional programming methods to machine learning approaches.

The purpose of this project is to develop an application that can predict if visiting a given website is safe or unsafe and let the final decision of opening the website to the user.

2. LITERATURE SURVEY

2.1. Existing Problem:

Web Phishing is one among the most common forms of cybercrime. The Internet Crime Report (2020) by FBI reports it to be the most common attack performed by cybercriminals. Phishing is a subset of a bigger school of thought called “Social Engineering”, which is the manipulation of people into believing the attacker. Phishing is done normally by hosting fake websites, sending fake emails, and conducting fake surveys. Phishing normally results in losing access to the user’s accounts but sometimes, the attacker stays silent to get even more information.

2.2. References:

1. A Survey and Classification of Web phishing detection schemes

Authors: Gaurav Varshney, Manoj Misra, Pradeep K. Atrey
Source: Wiley Online Library.com

Published on: 26 October 2016

Phishing is a fraudulent technique that is used over the Internet to deceive users with the goal of extracting their personal information such as username, passwords, credit card, and bank account information. The key to phishing is deception. Phishing uses email spoofing as its initial medium for deceptive communication followed by spoofed websites to obtain the needed information from the victims. Phishing was discovered in 1996, and today, it is one of the most severe cybercrimes faced by Internet users. Researchers are working on the prevention, detection, and education of phishing attacks, but to date, there is no complete and accurate solution for thwarting them.

2. Web Phishing Detection Using a Deep Learning Framework

Authors: Tony T. Luo, Ting Zhu, Wei Wang, Futai Zou
Source: Hindawi.com

Published on: 26 September 2018

Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet. Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity. It will lead to information disclosure and property damage. This paper mainly focuses on applying a deep learning framework to detect phishing websites. This paper first designs two types of features for web phishing: original features and interaction features. A detection model based on Deep Belief Networks (DBN) is then presented. The test using real IP flows from ISP (Internet Service Provider) shows that the detecting model based on DBN can achieve an approximately 90% true positive rate and 0.6% false positive rate.

3. Intelligent Web-Phishing detection and protection scheme using integrated features of images, frames and text

Authors: Moruf A Adebowale, M Alamgir Hossain

Source: scienceDirect.com

Received 25 April 2018, Revised 29 July 2018, Accepted 29 July 2018, Available online 4 August 2018 A phishing attack is one of the most significant problems faced by online users because of its enormous effect on the online activities performed. In recent years, phishing attacks continue to escalate in frequency, severity and impact. Several solutions, using various methodologies, have been proposed in the literature to counter the web-phishing threats. Notwithstanding, the existing technology cannot detect the new phishing attacks accurately due to the insufficient integration of features of the text, image and frame in the evaluation process. The use of related features of images, frames and text of legitimate and non-legitimate websites and associated artificial intelligence algorithms to develop an integrated method to address these together.

4. Systemation of Knowledge: A systematic review of software-based web phishing detection

Authors: Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha and M. Guizani.
Publisher: "Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection," in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2797-2819, Fourth quarter 2017

Phishing is a form of cyber-attack that leverages social engineering approaches and other sophisticated techniques to harvest personal information from users of websites. The average annual growth rate of the number of unique phishing websites detected by the Anti-Phishing Working Group is 36.29% for the past six years and 97.36% for the past two years. In the wake of this rise, alleviating phishing attacks has received a growing interest from the cyber security community. Extensive research and development have been conducted to detect phishing attempts based on their unique content, network, and URL characteristics.

5. Web Phishing Detection Based on Page Spatial Layout Similarity

Publisher: Weifeng Zhang School of Computer, Nanjing University of Posts and Telecommunications, China. Hua Lu Department of Computer Science, Aalborg University, Denmark

Web phishing is becoming an increasingly severe security threat in the web domain. Effective and efficient phishing detection is very important for protecting web users from loss of sensitive private information and even personal properties. One of the keys of phishing detection is to efficiently search the legitimate web page library and to find those page that are the most similar to a suspicious phishing page.

2.3. Problem Statement Definition:

Phishing is the act of accessing the private information of a user without their consent or knowledge. Everyone online is susceptible to a phishing attack, some of the common cases are as follows:

1) Users on social media platforms opening a URL which was sent to them with a false intent like giving rewards.
2) Students who are online in search of assignment answers and course material accessing a website which has what they want but at the same time steal information discreetly.
3) Users using malicious payment gateways for online transactions.

The goal of this project is to warn users of malicious URLs that they come across while surfing the web.

3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas:

Empathy Map is a widely used design practice which helps developers get into the minds of end users for a product by analysing what they may Say, Think, Do and Feel. Following is the Empathy Map the team came up with.

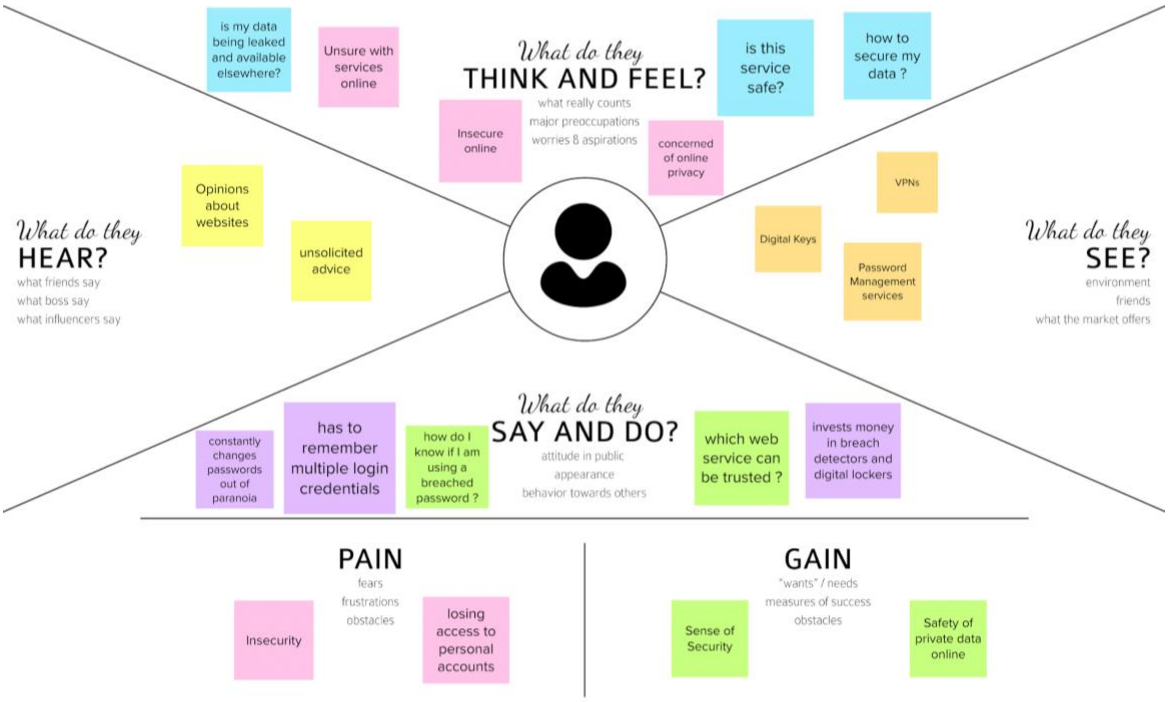

Fig 1. Empathy Map

3.2. Ideation and Brainstorming:

Fig 2. Brainstorm map (left) and Group Ideas (right)

3.3. Proposed Solution:

1) Train a machine learning model that can predict if a website is safe or unsafe and give a quantitative score for the same

2) Build a webpage for the user to give input to the proposed model and use the same to display the result.

3.4. Problem Solution Fit:

Fig 3. Website Phishing Detection Solution fit

4. REQUIREMENT ANALYSIS:

4.1. Functional Requirements:

Functional requirements are product features that developers must implement to enable the users to achieve their goals. They define the basic system behavior under specific conditions. For our work the Functional requirements are as follows:

Functional Requirement No.	Functional Requirement (Epic)	Sub Requirement (Sub tasks)
FR-1	**User Input**	Using web scraping, the URL which needs to be checked is given as input by the user automatically.
FR-2	**Feature Extraction**	Appropriate Expressions of the URL are extracted using Regular Expressions logic using programming and fed as features
FR-3	**Prediction**	Models must use classification based ML algorithms like Decision Tree, Random Forest Classifier, SVMs etc.
FR-4	**Verify the link provided by the user**	User inputs the link to be verified
FR-5	**Display the result**	If the site link is a phishing site, user must be aware and read the precautions displayed If the site link is legit, exit the application
FR-6	**Sharing and clearing the Queries**	If any doubts, send query Read FAQs

4.2. Non – Functional Requirements:

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. For our work the Non - Functional requirements are as follows:

Non-functional Requirement No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Engage the user about the process to ensure that the functionality can meet design and usability requirements. It relates to overall satisfaction of the user.
NFR-2	**Security**	Users need to be protected from malicious attacks when using the site.
NFR-3	**Reliability**	It focuses on preventing failures during the lifetime of the product or system, from commissioning to decommissioning.
NFR-4	**Performance**	It is the ability of the application to always run acceptably. In time-critical scenarios, even the smallest delay in processing data can be unacceptable.
NFR-5	**Availability**	Fault tolerance ensuring that the application can meet its availability targets to be resilient
NFR-6	**Scalability**	It is the ability for the application to scale to meet increasing demands; for example, at peak times or as the system becomes more widely adopted.

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 shows how data enters and leaves the system, what changes the information, and where data is stored. Following is the DFD proposed for the application.

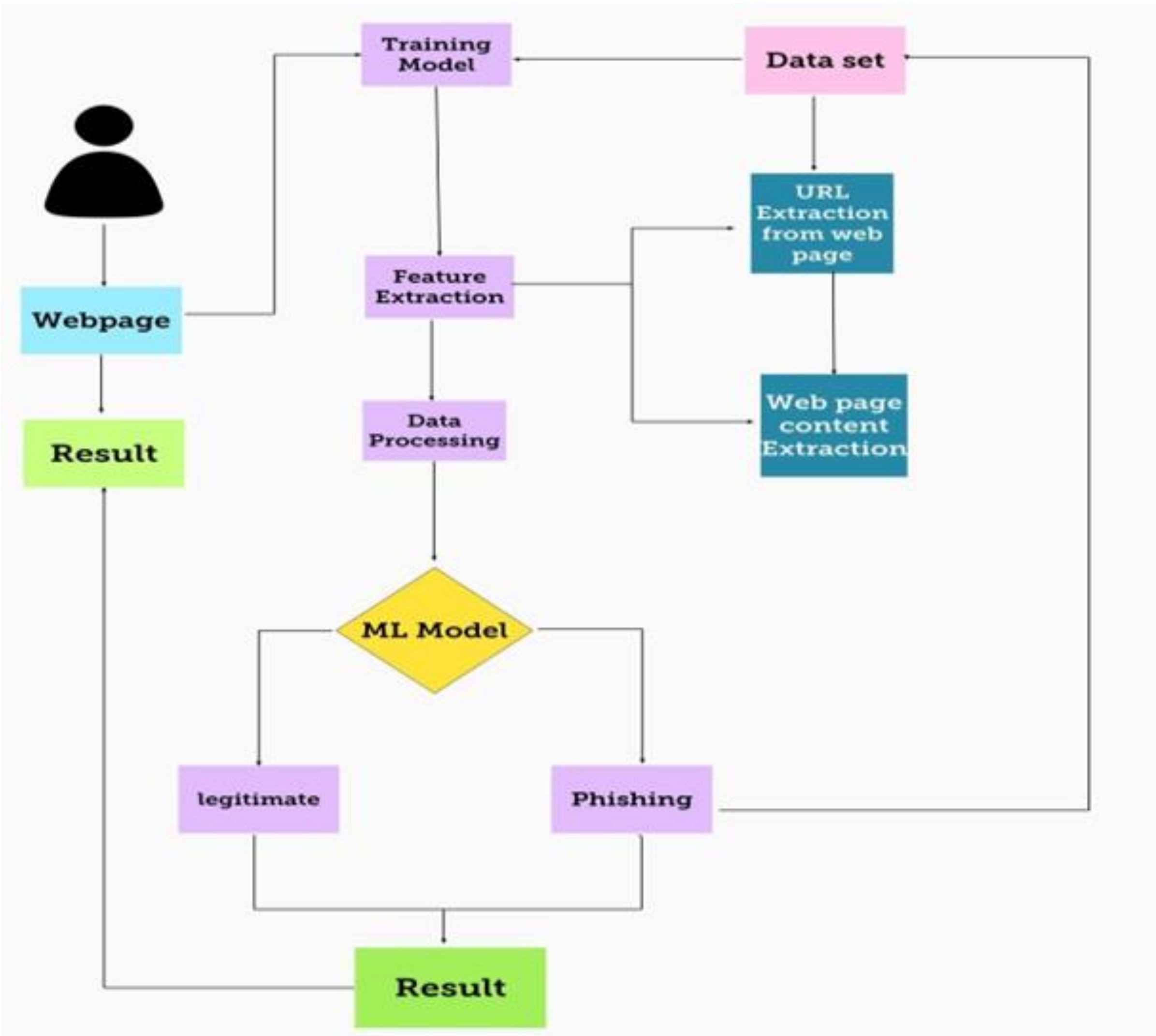

Fig 4. Data Flow Diagram

5.2. Solution and Technical Architecture:

Solution architecture is the process of developing solutions based on predefined processes, guidelines and best practices with the objective that the developed solution fits within the enterprise architecture in terms of information architecture, system portfolios, integration requirements and many more. Following is the proposed technical architecture along with the underlying components and technologies listed out.

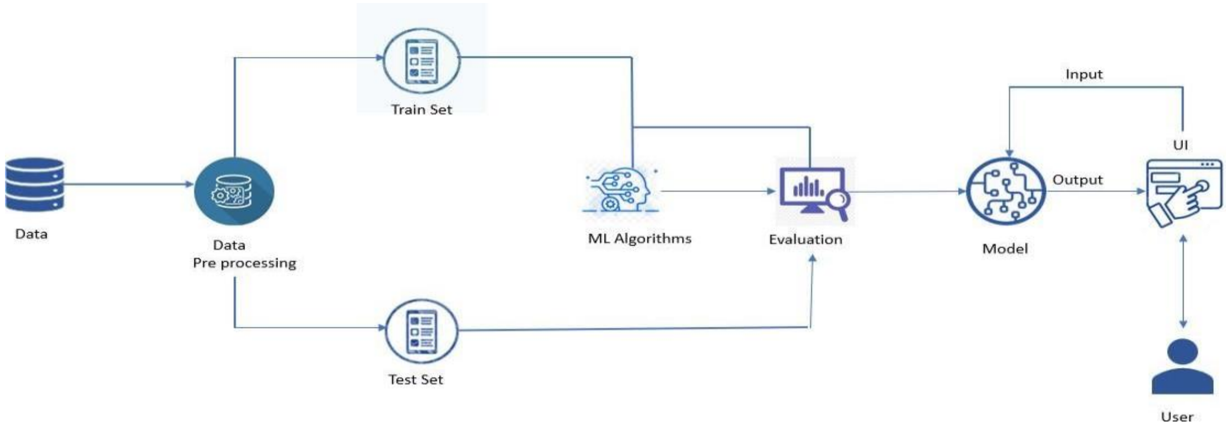

Fig 5. Technical Architecture

The Components and Technologies used in our work is tabulated below:

S. No	Component	Description	Technology
1	**User Interface**	Web Application, Cloud UI	HTML, CSS
2	**Application Logic-1**	Machine Learning Algorithms. Python Flask Application for Web App	Java / Python
3	**Database**	Stored Procedure (EXEC)	MySQL, NoSQL
4	**Cloud Database**	Database Service on Cloud	IBM DB2, IBM Cloudant
5	**File Storage**	File storage requirements	IBM Block Storage

Application Characteristics:

S. No	Characteristics	Description	Technology
1	**Open-Source Frameworks**	Gophish is a powerful, open-source phishing framework.	Machine Learning
2	**Security Implementations**	Encryption techniques and security algorithms.	AES 256, Cofense PDR
3	**Scalable Architecture**	Responsive UI/UX	React Framework, jQuery, Bootstrap, Cloudflare
4	**Availability**	Available at NLP, Spam Detection, Blacklisting or Reporting	Acunetix, Intruder, Ghost Phisher
5	**Performance**	Deployed and tested with multiple algorithms and this system gives greater accuracy and better performance than others.	Deep Learning

5.3. User Stories:

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer. Note that "customers" don't have to be external end users in the traditional sense, they can also be internal customers or colleagues within your organization who depend on your team. User stories are a few sentences in simple language that outline the desired outcome. They don't go into detail. Requirements are added later, once agreed upon by the team.

User Type	Functional Requirement	User Story/Task	Acceptance criteria
Customer (Mobile User)	**Registration**	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account/dashboard

		As a user, I will receive a confirmation email once I have registered for the application.	I can receive confirmation email and confirmation
		As a user, I can register for the application through Facebook.	I can register and access the dashboard with Facebook Login
		As a user, I can register for the application through Gmail.	I can register and access the dashboard with Gmail Login
	Login	As a user, I can log into the application by entering email & password	
Dashboard			
Customer (Web User)	**User Input**	As the user I can input the URL in the required field and waiting for a validation	I can go access the website without any problem
Customer Care Executive	**Feature Extraction**	After I compare in case if none found on comparison then we can extract feature using heuristic and visual similarity	In this I can have comparison between websites for security
Administrator	**Prediction**	Model will predict the URL websites using Machine Learning algorithms	In this I can have correct prediction on the algorithms
	Classifier	Here, I will send all the model output to classifier to produce final result	In this I will find the correct classifier for producing the result

6. PROJECT PLANNING AND SCHEDULING

6.1. Sprint Planning and Estimation:

A sprint is a set period where an agile team works to complete a specific set of development tasks. In most cases, there are multiple sprints within a larger development project. Sprints ultimately provide a framework for taking large, complex software projects and breaking them down into digestible phases. When a sprint ends, the team shows their work to the project owner, who reviews it. If the project meets expectations, the team moves on to the next sprint. Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

Sprint	Functional Requirement	User Story Number	User Story / Task	Priority
1	**User Input**	USN-1	User inputs an URL in the required field to check its validation	High
1	**Website Comparison**	USN-2	Model compares the websites using the Blacklist and Whitelist approach.	High
2	**Feature Extraction**	USN-3	After comparison, if none is found on comparison then it extracts features using heuristic and visual similarity.	Low
2	**Prediction**	USN-4	Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN.	Medium
3	**Classifier**	USN-5	Model then displays whether the website is legal site or a phishing site	High
3	**Announcement**	USN-6	Model then displays whether the website is legal site or a phishing site	High
4	**Events**	USN-7	This model needs the capability of retrieving and displaying accurate results for a website.	High

6.2. Sprint Delivery Schedule:

A sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Sprint Release Date
1	20	6 Days	24 Oct 2022	29 Oct 2022	29 Oct 2022
2	20	6 Days	31 Oct 2022	05 Nov 2022	06 Nov 2022
3	20	6 Days	07 Nov 2022	12 Nov 2022	12 Nov 2022
4	20	6 Days	14 Nov 2022	19 Nov 2022	19 Nov 2022

6.3. Reports from JIRA:

Reports in Jira help teams analyze progress on a project, track issues, manage their time, and predict future performance. They offer critical, real-time insights for Scrum, Kanban, and other agile methodologies, so that data-driven decisions can be made (the very best kind).

Fig 6. Reports from JIRA

7. CODING AND SOLUTIONING

7.1. Feature Selection:

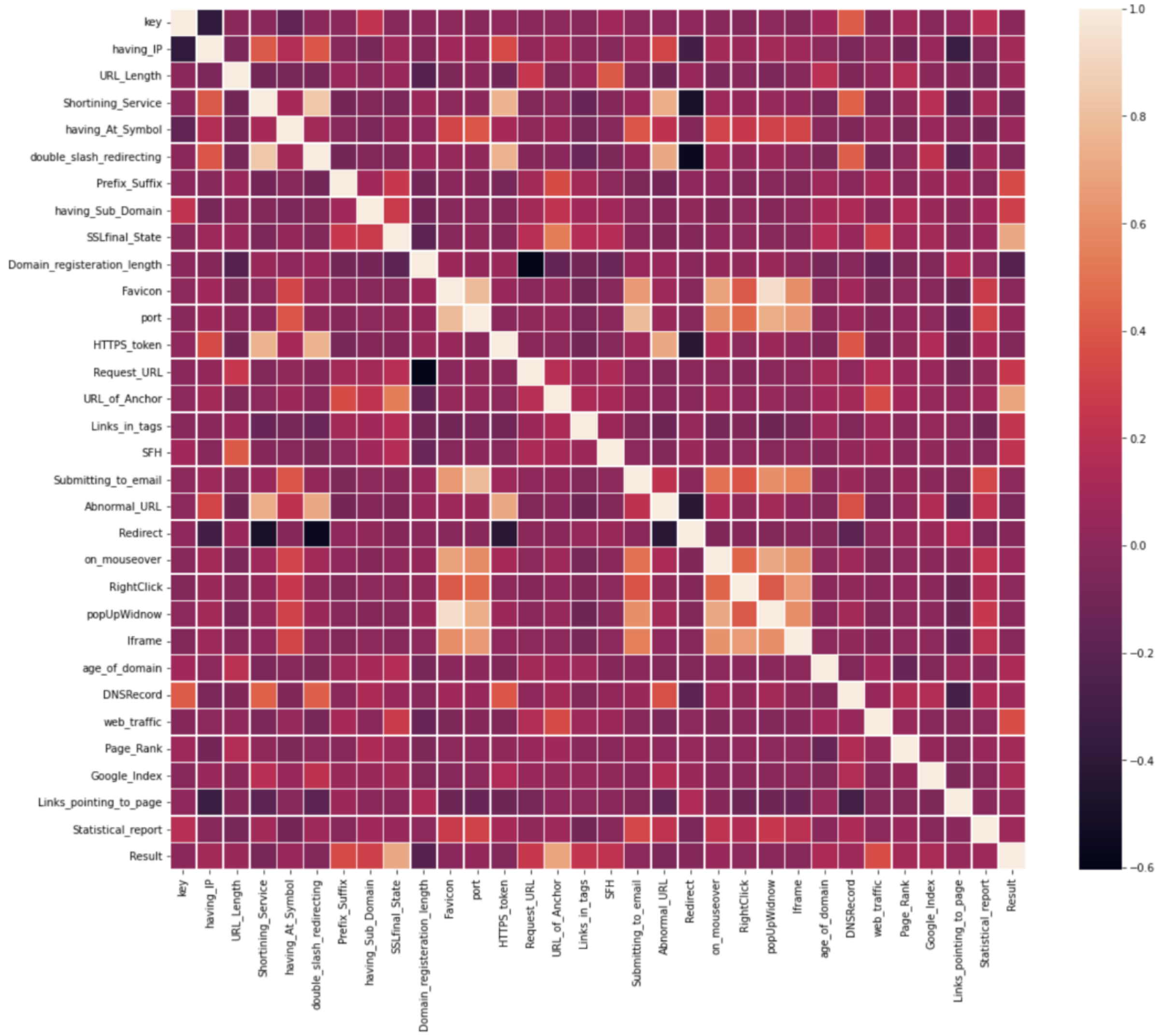

Fig 7. Correlation between features from the dataset

Selected Features: 'double_slash_redirecting', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor', 'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'popUpWidnow', 'Iframe'

7.2. Flask App

```
#importing required libraries

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

```
warnings.filterwarnings('ignore')
from feature_extraction import FeatureExtraction
file =
open("/Users/krishivijayanand/Downloads/Phishing_website.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)
```

7.3. Feature Extraction

```
import ipaddress
import re
from urllib import response
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|t
inyurl|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\.l
y|bit\.ly|ity\.im|'

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

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

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


    # 5.Redirecting//

    def redirecting(self):

        if self.url.rfind('//')>6:

            return -1

        return 1


    # 6.prefixSuffix

    def prefixSuffix(self):

        try:

            match = re.findall('\-', self.domain)

            if match:

                return -1

            return 1

        except:

            return -1


    # 7.SubDomains

    def SubDomains(self):

        dot_count = len(re.findall("\.", self.url))

        if dot_count == 1:

            return 1
```

```
        elif dot_count == 2:
            return 0
        return -1

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

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

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

```
            return -1
        except:
            return -1

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

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

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

```
        except:
            return -1


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


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


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


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

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

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

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

```
        except:
            return -1

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

            for link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', 
link['href'])]
                if self.url in link['href'] or self.domain in 
link['href'] or len(dots) == 1:
                    success = success + 1
                i = i+1
            for script in self.soup.find_all('script', src=True):
                dots = [x.start(0) for x in re.finditer('\.', 
script['src'])]
                if self.url in script['src'] or self.domain in 
script['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1
            try:
                percentage = success / float(i) * 100
                if percentage < 17.0:
                    return 1
                elif((percentage >= 17.0) and (percentage < 81.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1
```

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

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

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

```
        except:
            return -1

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

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

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

```
            return -1

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

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

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

            today  = date.today()
```

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

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

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

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

```
        except :
            return -1


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


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




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


    # 29. LinksPointingToPage
    def LinksPointingToPage(self):
        try:
            number_of_links = len(re.findall(r"<a href=", 
self.response.text))
            if number_of_links == 0:
```

```
                return 1
            elif number_of_links <= 2:
                return 0
            else:
                return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(
'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.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\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|'

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

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

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.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\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
            if url_match:
                return -1
```

```
            elif ip_match:
                return -1
            return 1
        except:
            return 1

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

7.4. Model Building:

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

from google.colab import drive
drive.mount('/content/drive')

train = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Phishing/Phising_Training_Dataset.csv')
test = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/Phishing/Phising_Testing_Dataset.csv')

print(train['Result'].value_counts())
plt.figure(figsize=(10, 7))
sns.countplot(train['Result'])

plt.figure(figsize=(18, 15))
sns.heatmap(train.corr(), linewidths=.5)

plt.figure(figsize=(8, 15))
heatmap =
sns.heatmap(train.corr()[['Result']].sort_values(by='Result',
ascending=False), vmin=-1, vmax=1, annot=True, cmap = 'viridis')
```

```
heatmap.set_title('Features Correlating with Result',
fontdict={'fontsize':18}, pad=16);

plt.savefig('heatmapfeaturecorr.png', dpi=300, bbox_inches='tight')


#clustermapping the dataframe correlations

plt.figure(figsize=(21, 18))

sns.clustermap(train.corr(),cmap='viridis')


#bar plot of correlation with Result label makes it easy to
understand dependencies

plt.figure(figsize=(10,8))

train.corr()["Result"][:-1].sort_values().plot(kind='bar')


X = train.drop(['key', 'Result'], axis=1)

y = pd.DataFrame(train['Result'])


cor_matrix = train.corr().abs()


upper_tri =
cor_matrix.where(np.triu(np.ones(cor_matrix.shape),k=1).astype(np.bo
ol))


sel_feature = [column for column in upper_tri.columns if
any(upper_tri[column] > 0.5)]

sel_feature


X = train[['double_slash_redirecting', 'port', 'HTTPS_token',
'Request_URL', 'URL_of_Anchor', 'Submitting_to_email',
'Abnormal_URL', 'Redirect',

           'on_mouseover', 'popUpWidnow', 'Iframe']]


X = pd.DataFrame(X)

y = pd.DataFrame(train['Result'])


from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state = 3, test_size = 0.3)


test = test[['double_slash_redirecting', 'port', 'HTTPS_token',
'Request_URL', 'URL_of_Anchor', 'Submitting_to_email',
'Abnormal_URL', 'Redirect',
           'on_mouseover', 'popUpWidnow', 'Iframe']]


# add grid search stuff
from sklearn.model_selection import GridSearchCV
param_grid={'C':[0.1,1,10,100,1000],'gamma':[1,0.1,0.01,0.001,0.0001
]}
grid = GridSearchCV(SVC(),param_grid,verbose=3)
grid.fit(X_train,y_train)


from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', C = 100.0, random_state = 0)
classifier.fit(X_train, y_train)


y_pred = classifier.predict(test)
print(y_pred)


PREDICTIONS_SVM = pd.DataFrame(y_pred,
columns=['Result']).to_csv('prediction_svm1.csv')


preds = pd.read_csv('/content/prediction_svm1.csv')


from collections import Counter
classes=Counter(preds1["Result"].values)
class_dist=pd.DataFrame(classes.most_common(),columns=["Class","Num_
of_Observations"])


from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

```
predictions = dtree.predict(X_test)

PREDICTIONS_DT = pd.DataFrame(predictions,
columns=['Result']).to_csv('prediction_dt.csv')

from sklearn.linear_model import LogisticRegression
lm=LogisticRegression()
lm.fit(X_train,y_train)

pred_lr=lm.predict(X_test)

PREDICTIONS_LR = pd.DataFrame(pred_lr,
columns=['Result']).to_csv('prediction_lr.csv')

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=30)
knn.fit(X_train,y_train)

pred_knn = knn.predict(X_test)

PREDICTIONS_KNN = pd.DataFrame(pred_knn,
columns=['Result']).to_csv('prediction_knn.csv')

from keras.layers import CuDNNLSTM, Dense, Dropout, LSTM
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow import keras
from tensorflow.keras.models import *

#using .map function to change -1 values to 0
train['Result'] = train['Result'].map({-1:0, 1:1})
train['Result'].unique()
```

```
X = train.drop('Result', axis=1)

y = pd.DataFrame(train['Result'])


from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state = 3, test_size = 0.4)


from sklearn.model_selection import train_test_split

X_test, X_val, y_test, y_val = train_test_split(X_test, y_test,
random_state = 3, test_size = 0.5)


model = Sequential()

model.add(LSTM(units = 8, activation = 'relu', input_shape = (None,
1)))

model.add(Dense(units = 1))

model.compile(optimizer = 'adam', loss = 'binary_crossentropy')

model.fit(X_train, y_train, validation_data = (X_val, y_val),
batch_size = 10, epochs = 20)


train_perf = model.evaluate(X_train, y_train)

test_perf = model.evaluate(X_test, y_test)


pred = np.round(model.predict(test))


for i in range(pred.size):

    if pred[i] == 0:

        pred[i] = -1


PREDICTIONS_LSTM = pd.DataFrame(pred,
columns=['Result']).to_csv('prediction_lstm.csv')
```

8. TESTING

8.1. Test Cases

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status
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/	Should Display the Webpage	Working as expected	Pass
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
LoginPage_TC_OO 3	Functional	Home page	Verify whether the link is legitimate or not	Enter URL and click go Type or copy paste the URL 3. Check the website is legitimate or not 4. Observe the results	https://phishingshield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass
LoginPage_TC_OO 4	Functional	Home Page	Verify user is able to access the legitimate website or not	Enter URL and click go 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://phishingshield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass
LoginPage_TC_OO 5	Functional	Home Page	Testing the website with multiple URLs	1.Enter URL 2.Type or copy paste the URL to test 3.Check the website is legitimate 4.Continue if the website is secure or be cautious if it is not secure	1. https://avbalajee.github.io/welcome 2. totalpad.com 3. delgets.com	User can be able to identify the websites whether it is secure or not	Working as expected	Pass

8.2. User Acceptance Testing

UAT 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).

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

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

9. RESULTS

9.1. Results

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

- Support Vector Machine Classifier

	precision	recall	f1-score	support
-1	0.97	0.94	0.96	976
1	0.96	0.98	0.97	1235
accuracy			0.96	2211
macro avg	0.97	0.96	0.96	2211
weighted avg	0.96	0.96	0.96	2211

- Decision Trees

	precision	recall	f1-score	support
-1	0.95	0.95	0.95	976
1	0.96	0.96	0.96	1235
accuracy			0.96	2211
macro avg	0.96	0.96	0.96	2211
weighted avg	0.96	0.96	0.96	2211

- Logistic Regression

	precision	recall	f1-score	support
-1	0.94	0.91	0.92	976
1	0.93	0.95	0.94	1235
accuracy			0.93	2211
macro avg	0.93	0.93	0.93	2211
weighted avg	0.93	0.93	0.93	2211

- K Nearest Neighbors Method

	precision	recall	f1-score	support
-1	0.95	0.95	0.95	976
1	0.96	0.96	0.96	1235
accuracy			0.96	2211
macro avg	0.96	0.96	0.96	2211
weighted avg	0.96	0.96	0.96	2211

- RNN-LSTMs

```
f1_score on training Data: 0.985
f1_score on test Data: 0.985

Recall on training Data: 0.978
Recall on test Data: 0.544

precision on training Data: 0.993
precision on test Data: 0.544
```

Hyper Parameter Tuning:

- Using Grid Search to tune the **best performing model - SVC**:

```
# Support Vector Classifier model
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'gamma': [0.1],'kernel': ['rbf','linear']}

svc = GridSearchCV(SVC(), param_grid)

# fitting the model for grid search
svc.fit(X_train, y_train)
```

```
GridSearchCV(estimator=SVC(),
             param_grid={'gamma': [0.1], 'kernel': ['rbf', 'linear']})
```

10. ADVANTAGES AND DISADVANTAGES

10.1. Advantages

1. Improve on Inefficiencies of Secure Gateways and Phishing Awareness Training

   As increasingly-sophisticated phishing attacks, such as BEC, become more difficult to detect, even by trained security personnel. Thus there is an urgent need for the channel to provide customers with technology that not only strives to prevent intrusion, but can also help users after an attack has passed through the secure email gateway.

2. It Takes a Load off the Security Team

   Customers now have many tools on the market to enhance their email security. The best of these use artificial intelligence and machine learning to better identify some of the suspected threats. This not only improves security, but can significantly reduce the workloads of IT and security teams. According to a survey by Fidelis Cybersecurity, less than one in five organizations have a dedicated threat hunting team, and only half of those could handle more than eight investigations per day.

10.2. Disadvantages

1. Great computational costs, as the model is actively learning the URLs thrown.
2. Developed model must have high precision, higher than what is obtained in this use case, as the impact of sensitive information theft and stealth of malicious websites is very high.

11. CONCLUSION

The most important way to protect the user from phishing attacks is education about malicious software and awareness. Internet users must be aware of all the security tips which are given by experts. Every user must be trained to blindly follow the links to the websites where they have to send their sensitive information. It is essential to check the URL before entering the websites.

Given the dataset of Phishing websites,we have explored a variety of Machine learning architectures and Deep learning architectures to assess the website's trustability and protection level. The correlation values for each feature was obtained and the top 12 features were used for training.

Models used:

- Support Vector Machine Classifier
- Decision Trees
- Logistic Regression
- K Nearest Neighbors Method
- RNN-LSTMs

Highest accuracy was obtained while using the Support Vector Classifier (SVC). This produced us with a mean accuracy of 96% which is a massive improvement and covers all the basic requirements to make an abstract model of an anti-phishing website.

12. FUTURE SCOPE

In the future if we get a structured dataset of phishing we can perform phishing detection much faster than any other technique.Going forward,we can use a combination of any other two or more classifiers to get maximum accuracy. Such kinds of ensemble learning techniques could be really helpful in use cases like ours as one classifier can pick up a feature that the other classifier cannot.

We can also plan to explore various phishing techniques that use Lexical Features, Network based features, Content based features, Web Page based features and HTML and JavaScript features of web pages which can improve the performance of the system. In Particular, we extract features from URLs and pass it through the various classifiers.

13. APPENDIX

13.1. Source Code:

https://drive.google.com/drive/folders/1gz_EwIzpaIbu3yvKOxcPzAo5OCC_WbTR?usp=sharing

13.2. GitHub and Project Demo Link

- Github Link: https://github.com/IBM-EPBL/IBM-Project-27413-1660056032
- Project Demo Link: https://drive.google.com/file/d/1AtYQZKbsAHCXZi52Iq3F01rNvw8C4Jtu/view?usp=sharing