A Major Project Report

On

Machine Learning based Phishing Website URL Detection

Submitted to JNTU HYDERABAD

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted
By

P.Sai Rahul	**(178R1A05G4)**
G.Rahul Yadav	**(178R1A05D3)**
P.Sai Roshan	**(178R1A05H0)**
K.Rukmini Reddy	**(178R1A05E6)**

Under the Esteemed guidance of
Mrs. Y. Pratima
Assistant Professor, Department of CSE

Department of Computer Science & Engineering

CMR ENGINEERING COLLEGE
(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

2020-2021

CMR ENGINEERING COLLEGE

(Accredited by NBA,Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **“Machine Learning based Phishing Website URL Detection”** is a bonafide work carried out by

P.Sai Rahul	**(178R1A05G4)**
G.Rahul Yadav	**(178R1A05D3)**
P.Sai Roshan	**(178R1A05H0)**
K.Rukmini Reddy	**(178R1A05E6)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	Major Project Coordinator	Head of the Department
Mrs. K. Pratima	**Mrs. G. Sumalatha**	**Dr. Sheo Kumar**
Assistant Professor	Associate Professor	Professor & H.O.D
Department of CSE	Department of CSE	Department of CSE,
CMREC, Hyderabad	CMREC, Hyderabad	CMREC, Hyderabad

DECLARATION

This is to certify that the work reported in the present project entitled "**Machine Learning based Phishing Website URL Detection "** is a record of bonafide work done by us in the Department of Computer Science and Engineering, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

P.Sai Rahul	**(178R1A05G4)**
G.Rahul Yadav	**(178R1A05D3)**
P.Sai Roshan	**(178R1A05H0)**
K.Rukmini Reddy	**(178R1A05E6)**

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr.Sheo Kumar**, HOD, **Department of CSE, CMR Engineering College** for their constant support**.**

I am extremely thankful to **Mrs. Y. Pratima,** Assistant Professor, Internal Guide, Department of CSE, for his/ her constant guidance, encouragement and moral support throughout the project.

I will be failing in duty if I do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

I express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, I am very much thankful to my parents who guided me for every step.

P.Sai Rahul	**(178R1A05G4)**
G.Rahul Yadav	**(178R1A05D3)**
P.Sai Roshan	**(178R1A05H0)**
K.Rukmini Reddy	**(178R1A05E6)**

CONTENTS

ABSTRACT

There are a number of users who purchase products online and make payment through e-banking. There are e-banking websites who 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. This Guided Project mainly focuses on applying a machine learning algorithm to detect Phishing websites.

LIST OF FIGURES

1.INTRODUCTION

1.1 Introduction & Objectives

Phishing is a kind of Cybercrime trying to obtain important or confidential information from users which is usually carried out by creating a counterfeit website that mimics a legitimate website. Phishing attacks employ a variety of techniques such as link manipulation, filter evasion, website forgery, covert redirect, and social engineering. The most common approach is to set up a spoofing web page that imitates a legitimate website. These type of attacks were top concerns in the latest 2018 Internet Crime Report, issued by the U.S. Federal Bureau of Investigations Internet Crime Complaint Center (IC3). The statistics gathered by the FBIs IC3 for 2018 showed that internet-based theft, fraud, and exploitation remain pervasive and were responsible for a staggering $2.7 billion in financial losses in 2018. In that year, the IC3 received 20,373 complaints against business email compromise (BEC) and email account compromise (EAC), with losses of more than $1.2 billion. The report notes that the number of these sophisticated attacks have grown increasingly in recent years. Anti-Phishing Working Group(APWG) emphasizes that phishing attacks have grown in recent years, illustrates the total number of phishing sites detected by APWG in the first quarter of 2020 and the last quarter of 2019.

Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.
- It will lead to information disclosure and property damage.
- Large organizations may get trapped in different kinds of scams.

This Project mainly focuses on applying a machine learning algorithm to detect Phishing websites.

1.2 Purpose of the Project

We use machine learning to tell if a website url is legit or work of hacker to steal your private information such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.

1.3 Existing system & Disadvantages

List-based phishing detection systems use two list, whitelists and blacklists, for classifying the legitimate and phishing web pages. Whitelist-based phishing detection systems make secure and legitimate websites to provide the necessary information. Each website that is not in the whitelist is considered as suspicious. Cao et al. (2008) developed a system that creates a whitelist by registering the IP address of each site, which has Login user inter-face that the user has visited. When the user visits a website, the system will warn if there is an incompatibility in the registered in-formation of the website. However, this method is considered suspect in the legitimate sites visited by the user for the first time. Jain and Gupta (2016) developed a method that warns users on the web with a white-list of legitimate websites that is updated automatically. The method consists of two phases, the domain - IP address matching module and the extraction of the features of the links in the source code. According to the experimental results, 86.02% true positive rate and 148% false negative rate were achieved in this work. Blacklists are created by URL records, which are known as phishing websites. These list entries are derived from a number of sources, such as spam detection systems, user notifications, third-party organizations, etc. The use of blacklists makes it impossible for attackers to attack again via same URL or IP address, which are previously used for attack. The security mechanism updates black- lists either by detecting malicious URLs / IPs or users can download these lists instantly from a server and protect their systems against the attacks listed in this list. The blacklist-based systems, however, do not have the ability to detect an actual attack or a first-time attack (zero-day attack). These attack detection mechanisms have a lower false positive rate than machine learning based systems. The success of the blacklist-based phishing attack detection system is about 20% (Khonji, Iraqi, & Jones, 2013; Sheng, Holbrook, Kumaraguru, Cranor, & Downs, 2010). Therefore, it seems that blacklist-based systems are not efficient as a reliable attack detection mechanism. Some companies service blacklist-based phishing attack detection systems such as Google Safe Browsing API (Google Safe Browsing, 2012), Phish Net (Prakash, Kumar, Kompella, & Gupta, 2010). These systems use an approximate matching algorithm to check whether the suspicious URL exists in the blacklist or not. The blacklist-based approaches require frequent updates. In addition, the rapid growth of the blacklist requires excessive system resources (Sharifi, & Siadati, 2008; Sheng et al., 2009). Apart from the static techniques, dynamic techniques, which can learn from the previous data (especially big data) can produce a better solution with the help of machine learning approaches.

1.4 Proposed system with Features

In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not. One of the popular methods of malicious websites' detection is the use of machine learning methods. Mainly, detection of phishing attack is a simple classification problem. In order to develop a learning-based detection system, training data must contain lots of features, which are related to phishing and legitimate website classes. By the use of a learning algorithm, it can be easy to detect the unseen or not classified URLs with a dynamic mechanism. A text-based phishing detection technique was developed by Zhang and named as CANTINA, which extracts keywords using the term frequency-inverse document frequency algorithm (Zhang, Hong, & Cranor, 2007). Then, the keywords were searched by Google search engine. If the website was included in search results, it was classified as legitimate. However, the success of the study is limited because it is sensitive only to the English language. The enhanced model is named as CANTINA + , and it includes 15 HTML-based attributes (Xiang, Hong, Rose, & Cranor, 2011). The system achieved a 92% accuracy rate, but it could produce a large number of false positives. Tan, Chiew, Wo ng, and Sze (2016) developed an anti-phishing system named PhishWHO, which has three steps to identify a website legitimate or not. At the first stage, key- words are identified from the suspicious website. In the second stage, these keywords are used for the detection of possible target domains by a search engine. The target domain is identified using the features extracted from these websites. Finally, the system decides whether the website queried in the third step is legitimate or not.

Advantages

- **Language independence:** In most of the phishing detection system, language is very critical for the execution of the system. However, in the proposed system, we are using only URLs whose texts are constructed with random and long strings, which contain some specific keywords in a vector structure. Therefore, the success of our system depends on this word vector, and this can be constructed in a language independent way.
- **Huge Size of Phishing and Legitimate Data:** Constructing a dataset for the anti-phishing system is a trivial issue. There are some web-based services, which give URLs of the phishing web pages.
    - However, they share a limited amount of data in their web pages. Therefore, we wrote a script to collect all these data with different criteria. This is relatively easy than collecting the legitimate web page addresses. These addresses can be reached from the search engines and some services, which share the most attacked sites. We have obtained these URLs by writing some scripts and construct our dataset, which wholly contains 36,400 legitimate URLs and 37,175 phishing URLs.
- **Real-time Execution:** Since creating a web page is a cheap and easy task, for phishing the users, an attacker can quickly construct a fraudulent web page, which is active in a short lifespan (maybe for a few hours). Therefore, detection of the phishing web page in real-time is essential for the prevention from this type of attacks. In the proposed system, features depend on the address of the web page, and some features are constructed with the help of NLP algorithms. Therefore, it is executed faster and classifies the web page in a negligible amount of time.

- **Detection of new Websites:** Due to NLP based and word vector features, the proposed system can detect new phishing websites, which are not labeled as phishing previously. With this property, our system is robust for the zero-day attack, which is one of the most dangerous attack types in phishing.
- **Independence from Third-Party Services:** In the literature, there are many works, which use third-party services such as who is records, web-based blacklist/whitelist, ranking pages, network traffic measures, the age of domain detection, etc. Mainly, use of these services increases the efficiency of the detection/prevention system. However, if the aim is about the execution in the realtime, then these services increase the detection time; therefore, they cannot be useful.
- **Use of Feature-Rich Classifiers:** In the general form of featurebased machine learning algorithms, the first aim is to find a set of discriminative features, which can help to differentiate the phishing and legitimate web pages. After that, an appropriate machine learning model is executed. Most of the proposed detection/prevention systems use the limited number of features between 8 and 50 as depicted in Table 1. Obtaining a high-quality feature set is very crucial for the effectiveness of the system. There are lots of features in the literature, and some of them are not distinguishing enough. Therefore, we try to increase the number of features that can be used in our system and then eliminate the ones, which are not discriminative enough. We used 40 NLP based features and 1701 Word Features, and then this number is decreased to the 102 features with a feature reduction mechanism.

2. LITERATURE SURVEY

- Wong, R. K. K. (2019). An Empirical Study on Performance Server Analysis and URL Phishing Prevention to Improve System Management Through Machine Learning. In Economics of Grids, Clouds, Systems, and Services: 15th International Conference, GECON 2018, Pisa, Italy, September 18-20, 2018, Proceedings (Vol. 11113, p. 199). Springer.

- Rao, R. S., & Pais, A. R. (2019). Jail-Phish: An improved search engine based phishing detection system. Computers & Security, 83, 246-267.

- Ding, Y., Luktarhan, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. computers & security, 84, 256-275.

- Marchal, S., Saari, K., Singh, N., & Asokan, N. (2016, June). Know your phish: Novel techniques for detecting phishing sites and their targets. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 323-333). IEEE.

- Shekokar, N. M., Shah, C., Mahajan, M., & Rachh, S. (2015). An ideal approach for detection and prevention of phishing attacks. Procedia Computer Science, 49, 82-91.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 Problem Specification

This Project focuses on applying a machine learning algorithm to detect Phishing websites.

3.2 Modules and their Functionalities

This project provides the following Module:

User Module: User can open the website and enter any url for which he/she wants to find whether if the url is legit or work of the hackers.

3.3 Functional Requirements

1.Speed.

2. User Friendliness.

3. Ease of use in checkout flow.

4. Personalization.

5.Accessibility.

3.4 Non-Functional Requirements

1.Security

2.Privacy

3.Scalability and performance

4.Speed of key user journeys

5.Speed of web services

6.Accessibility

3.5 Feasibility Study

How feasible is the system proposed? This was analyzed by comparing the following factors with both the existing system and proposed system.

Cost: The cost required in the proposed system is comparatively less to the existing system.

Effort: the proposed system will provide a better working environment in which there will be ease of work and the effort required will be comparatively less than the existing system.

Time: Also, the time required generating a report or for doing any other work will be comparatively less**.**

4. SOFTWARE & HARDWARE REQUIREMENTS

4.1 Software requirements

Operating Systems: Windows 7 , 8 and 10

IDE: Pycharm

Programming Languages: Python

Framework: Flask

Software/Packages: Anaconda Navigator, Sklearn, NumPy, Pandas, Matplotlib

4.2 Hardware requirements

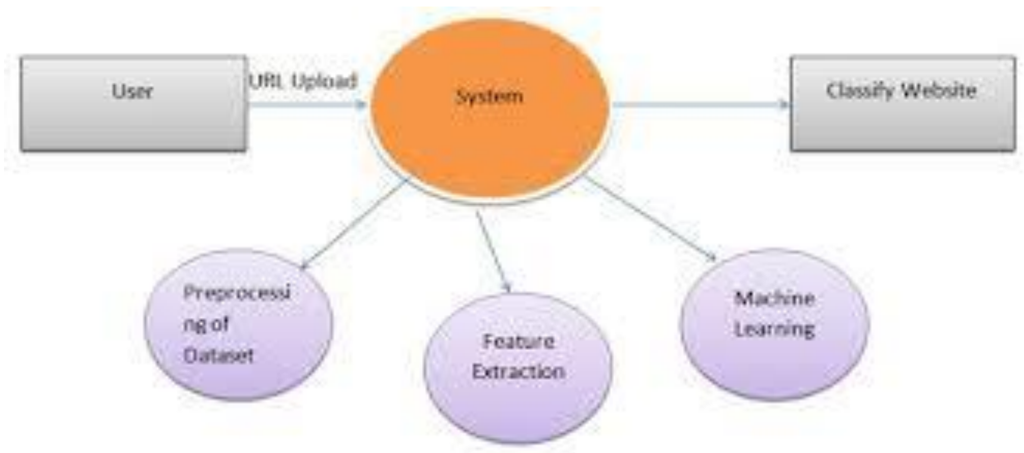
Processor	-	Pentium –III
Speed	-	1.1 Ghz
RAM	-	256 MB(min)
Hard Disk	-	20 GB
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

5. SOFTWARE DESIGN

5.1 Data Flow Diagram

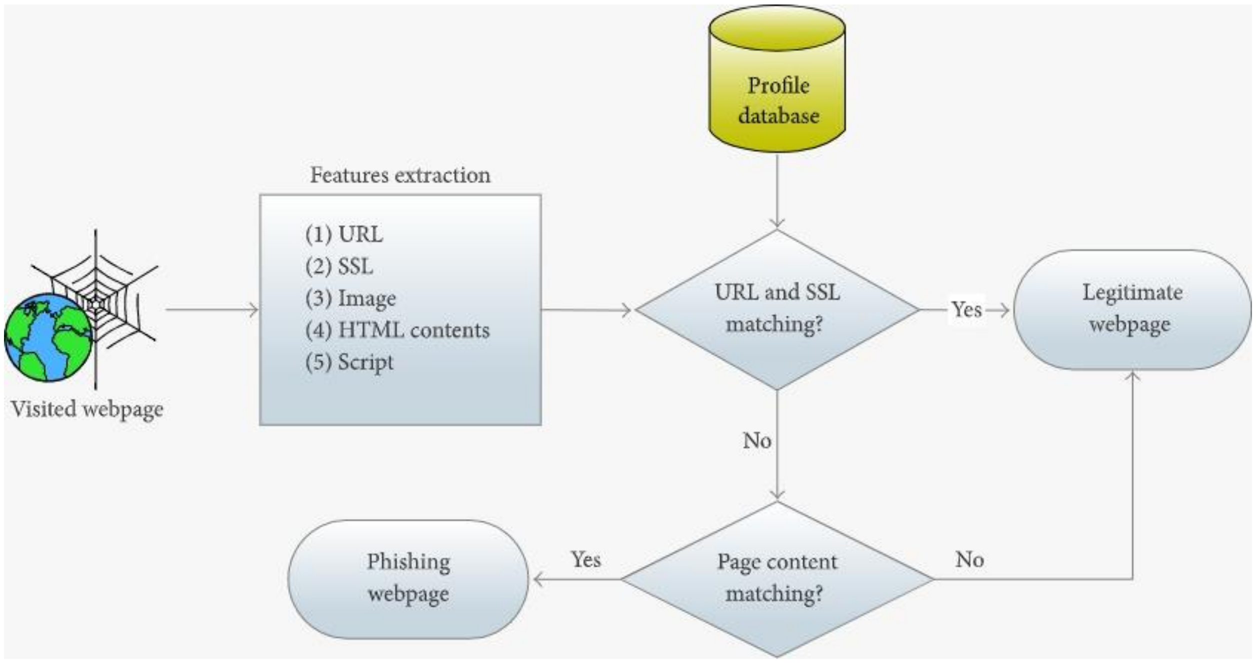
A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself.

LEVEL 1 DFD:



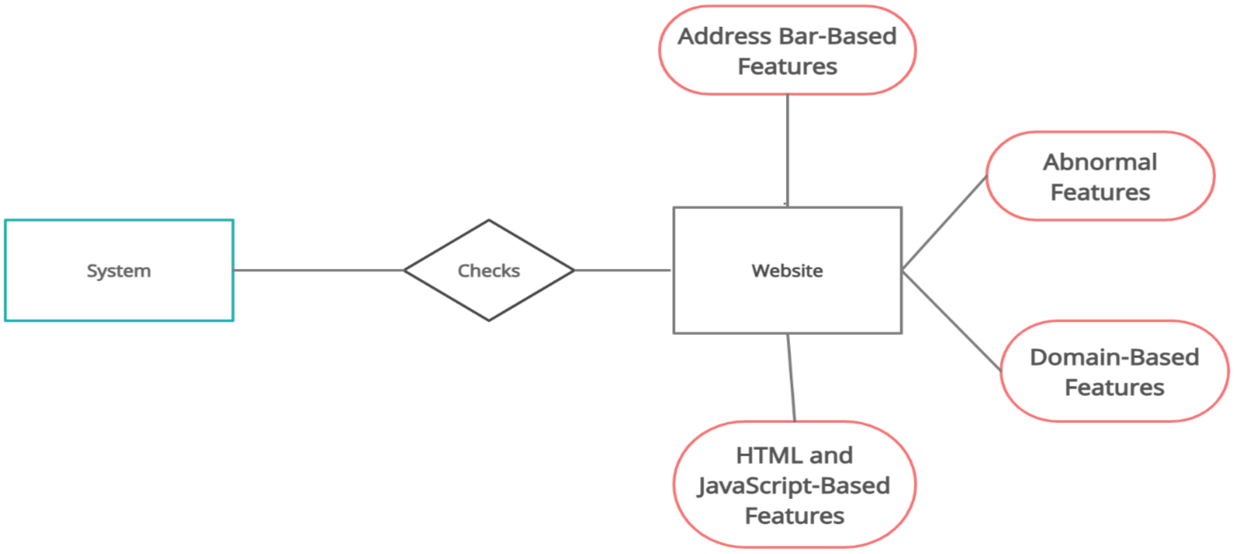
5.2 Control Flow Diagrams

It is a graphical representation of control flow or computation during the execution of programs or applications.



5.3 ER Diagrams

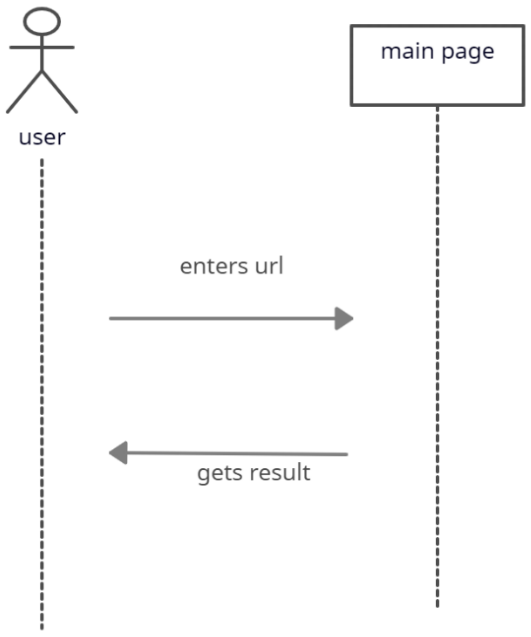
It describes the structure of a database with the help of a diagram called as ER diagram.


Address Bar-Based Features
Abnormal Features
System
Checks
Website
Domain-Based Features
HTML and JavaScript-Based Features

5.4 UML Diagrams

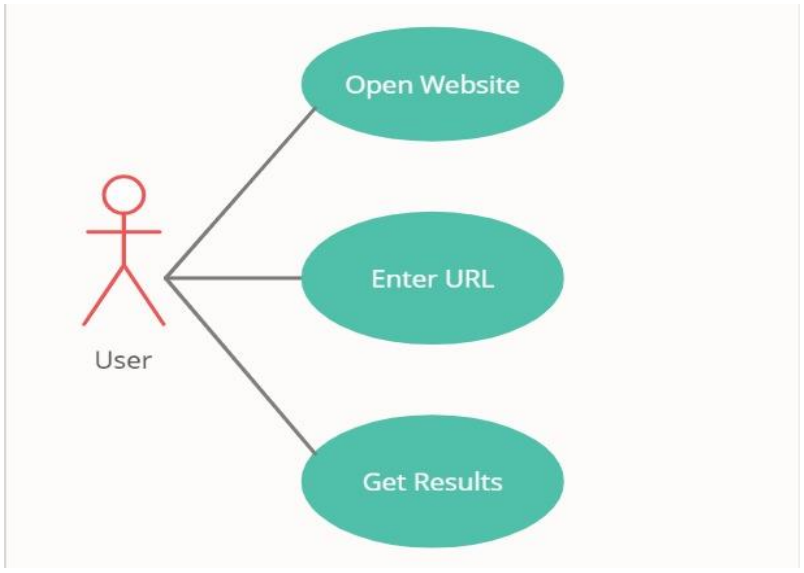
Sequence Diagram:

It shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



Use Case Diagram:

Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.



6. CODING AND IMPLEMENTATION

6.1Implementation

1. In order to develop this project we need to install following softwares/packages:

Anaconda Navigator:

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

2. To build Machine learning models you must require the following packages

Sklearn: Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.

NumPy: NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object

Pandas: pandas is a fast, powerful, flexible, and easy to use open source data analysis and manipulation tool,built on top of the Python programming language.

Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Flask: Web framework used for building Web applications

If you are using **anaconda navigator**, follow below steps to download required packages:

- Open anaconda prompt.
- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

6.2 CODE:

App.py :

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
#importing the inputScript file used to analyze the URL
import inputScript


#load model
app = Flask(__name__)
model = pickle.load(open('Phishing_Website.pkl', 'rb'))


#Redirects to the page to give the user iput URL.
@app.route('/predict')
def predict():
    return render_template('final.html')

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    url = request.form['URL']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="Your are safe!!  This is a Legitimate Website."

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('final.html', prediction_text='{}'.format(pred),url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

inputscript.py :

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

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

#Checking for the URL length. Returns 1 (Legitimate) if the URL length is less than 54
characters
```

```
#Returns 0 if the length is between 54 and 75
#Else returns -1;
def URLURL_Length (url):
    length=len(url)
    if(length<=75):
         if(length<54):
             return 1
         else:
             return 0
    else:
        return -1

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

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

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

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

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

#Checking for @ symbol. Returns 1 if no @ symbol found. Else returns 0.
def having_At_Symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return 1
    else:
        return -1

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

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

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

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

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

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

#Checking the Favicon. Returns 1 if the domain of the favicon image and the URL domain
match else returns -1.
def Favicon(url):
    subDomain, domain, suffix = extract(url)
    b=domain
    try:
        icons = favicon.get(url)
        icon = icons[0]
        subDomain, domain, suffix =extract(icon.url)
        a=domain
        if(a==b):
            return 1
        else:
            return -1
    except:
        return -1

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

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

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

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

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

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

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

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

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

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

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

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

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

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

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

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

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

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

    except:
        return 0


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

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

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

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

#using iframe returns -1, otherwise returns 1
```

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



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

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

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

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

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

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

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


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

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

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

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

ip_match=regex.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.11
6|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\
.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.1
08|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\
.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|1
0\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.1
04\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|2
08\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217
\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.2
14\.197\.72|87\.98\.255\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.
46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',
ip_address)
    except:
        return -1

    if url_match:
        return -1
    else:
        return 1

#returning scrapped data to calling function in app.py
def main(url):
    check = [[having_IPhaving_IP_Address
(url),URLURL_Length(url),Shortining_Service(url),having_At_Symbol(url),

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

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

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

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


    print(check)
    return check
```

index.html :

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Phishing Website Detection</title>
  <meta content="" name="description">
  <meta content="" name="keywords">




  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|
Jost:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600
i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="assets/vendor/icofont/icofont.min.css" rel="stylesheet">
  <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="assets/vendor/remixicon/remixicon.css" rel="stylesheet">
  <link href="assets/vendor/venobox/venobox.css" rel="stylesheet">
  <link href="assets/vendor/owl.carousel/assets/owl.carousel.min.css" rel="stylesheet">
  <link href="assets/vendor/aos/aos.css" rel="stylesheet">


          <!--link href="{{ url_for('static', filename='/vendor/bootstrap/css/bootstrap.min.css') }}"
rel="stylesheet">
          <link href="{{ url_for('static', filename='/vendor/icofont/icofont.min.css') }}"
rel="stylesheet">
          <link href="{{ url_for('static', filename='/vendor/boxicons/css/boxicons.min.css') }}"
rel="stylesheet">
          <link href="{{ url_for('static', filename='/vendor/remixicon/remixicon.css') }}"
rel="stylesheet">
          <link href="{{ url_for('static', filename='/vendor/venobox/venobox.css') }}"
rel="stylesheet">
          <link href="{{ url_for('static',
filename='/vendor/owl.carousel/assets/owl.carousel.min.css') }}" rel="stylesheet">
          <link href="{{ url_for('static', filename='/vendor/aos/aos.css') }}" rel="stylesheet"-->

  <!-- Template Main CSS File -->
  <link href="assets/css/style.css" rel="stylesheet">
  <!--link href="{{ url_for('static', filename='css/style.css') }}" rel="stylesheet"-->


</head>

<body>

  <!-- ======= Header ======= -->
  <header id="header" class="fixed-top ">
    <div class="container d-flex align-items-center">

      <h1 class="logo mr-auto"><a href="index.html">CMREC</a></h1>
      <!-- Uncomment below if you prefer to use an image logo -->
      <!-- <a href="index.html" class="logo mr-auto"><img src="assets/img/logo.png" alt=""
class="img-fluid"></a>-->

      <nav class="nav-menu d-none d-lg-block">
        <ul>
          <li class="active"><a href="index.html">Home</a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#contact">Contact</a></li>

        </ul>
      </nav><!-- .nav-menu -->

      <a href="http://localhost:5000/predict" class="get-started-btn scrollto">Get Started</a>

    </div>
  </header><!-- End Header -->

  <!-- ======= Hero Section ======= -->
  <section id="hero" class="d-flex align-items-center">

    <div class="container">
      <div class="row">
        <div class="col-lg-6 d-flex flex-column justify-content-center pt-4 pt-lg-0 order-2 order-lg-
1" data-aos="fade-up" data-aos-delay="200">
          <h1>Solution to Detect Phishing Websites      </h1>
          <h2>Be aware of what's happening with you confidential data</h2>
          <div class="d-lg-flex">
            <a href="http://localhost:5000/predict" class="btn-get-started scrollto">Get Started</a>

          </div>
        </div>
        <div class="col-lg-6 order-1 order-lg-2 hero-img" data-aos="zoom-in" data-aos-
delay="200">
          <img src="https://www.technologyvisionaries.com/wp-content/uploads/2020/01/bigstock-
Data-Phishing-Hacker-Attack-t-319270852-scaled.jpg" class="img-fluid animated" alt="">
        </div>
      </div>
    </div>

  </section><!-- End Hero -->

  <main id="main">



    <!-- ======= About Us Section ======= -->
    <section id="about" class="about">
      <div class="container" data-aos="fade-up">

        <div class="section-title">
          <h2>About</h2>
        </div>

        <div class="row content">
          <div class="col-lg-6">
            <p>
```

```
              Web service is one of the key communications software services for the Internet. Web
phishing is one of many security threats to web services on the Internet.  Web phishing aims to
steal private information, such as usernames, passwords, and credit card details, by way of
impersonating a legitimate entity.
            </p>

          </div>
          <div class="col-lg-6 pt-4 pt-lg-0">
            <p>
             The recipient is then tricked into clicking a malicious link, which can lead to the
installation of malware, the freezing of the system as part of a ransomware attack or the
revealing of sensitive information. It will lead to information disclosure and property damage.
            </p>

          </div>
        </div>

      </div>
    </section><!-- End About Us Section -->

            <!-- ======= Cta Section ======= -->
    <section id="cta" class="cta">
      <div class="container" data-aos="zoom-in">

        <div class="row">
          <div class="col-lg-9 text-center text-lg-left">
            <h3>Check your Website</h3>
            <p>Understanding if the website is a valid one or not is importand and plays a vital role
in the securing the data. To know if the URL is a valid one or you are information is at risk check
your website. </p>
          </div>
          <div class="col-lg-3 cta-btn-container text-center">
            <a class="cta-btn align-middle" href="http://localhost:5000/predict">Check your
website</a>
          </div>
        </div>

      </div>
    </section><!-- End Cta Section -->


    <!-- ======= Services Section ======= -->
    <section id="services" class="services section-bg">
      <div class="container" data-aos="fade-up">

        <div class="section-title">
```

```
          <h2>Protect yourself from Phishing Attacks</h2>
          <p>As a report from the Anti-Phishing Working Group (APWG) revealed earlier this year,
there has been a notable rise in the number phishing attacks. It’s a widespread problem, posing a
huge risk to individuals and organizations</p>
                          <p>Follow the tips below and stay better protected against phishing attacks.</p>
        </div>

        <div class="row">
                        <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-
aos="zoom-in" data-aos-delay="400">
            <div class="icon-box">
              <div class="icon"><i class="bx bx-layer"></i></div>
              <h4>Browse securely with HTTPs</h4>
              <p>You should always, where possible, use a secure website (indicated by https:// and a
security “lock” icon in the browser’s address bar) to browse, and especially when submitting
sensitive information online, such as credit card details.</p>
            </div>
          </div>
          <div class="col-xl-3 col-md-6 d-flex align-items-stretch" data-aos="zoom-in" data-aos-
delay="100">
            <div class="icon-box">

              <div class="icon"><i class="bx bxl-dribbble"></i></div>
              <h4>Watch out for shortened links</h4>
              <p>Cybercriminals often use these – from Bitly and other shortening services – to trick
you into thinking you are clicking a legitimate link, when in fact you’re being inadvertently
directed to a fake site.</p>
            </div>
          </div>

          <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-md-0" data-aos="zoom-
in" data-aos-delay="200">
            <div class="icon-box">
              <div class="icon"><i class="bx bx-file"></i></div>
              <h4>Does that email look suspicious? Read it again</h4>
              <p>Plenty of phishing emails are fairly obvious. They will be punctuated with plenty of
typos, words in capitals and exclamation marks.</p>
            </div>
          </div>

          <div class="col-xl-3 col-md-6 d-flex align-items-stretch mt-4 mt-xl-0" data-aos="zoom-
in" data-aos-delay="300">
            <div class="icon-box">
              <div class="icon"><i class="bx bx-tachometer"></i></div>
              <h4>Be wary of threats and urgent deadlines</h4>
              <p>Some of these threats may include notices about a fine, or advising you to do
something to stop your account from being closed. Ignore the scare tactics and contact the
company separately via a known and trusted channel.</p>
          </div>
        </div>




      </div>

    </div>
  </section><!-- End Services Section -->

  <!-- ======= Contact Section ======= -->
  <section id="contact" class="contact">
    <div class="container" data-aos="fade-up">

      <div class="section-title">
        <h2>Contact</h2>
                     <p>Team No : 13<p>
                     <p>Major Project, CSE Department, 2021<p>
        <p>CMR ENGINEERING COLLEGE</p>
      </div>
    </div>
  </section><!-- End Contact Section -->

</main><!-- End #main -->

<!-- ======= Footer ======= -->
<footer id="footer">
  <div class="footer-top">
    <div class="container">
      <div class="row">
      </div>
    </div>
  </div>

  <div class="container footer-bottom clearfix">

</footer><!-- End Footer -->

<a href="#" class="back-to-top"><i class="ri-arrow-up-line"></i></a>
<div id="preloader"></div>

<!-- Vendor JS Files -->
```

```
  <script src="assets/vendor/jquery/jquery.min.js"></script>
  <script src="assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="assets/vendor/jquery.easing/jquery.easing.min.js"></script>
  <script src="assets/vendor/php-email-form/validate.js"></script>
  <script src="assets/vendor/waypoints/jquery.waypoints.min.js"></script>
  <script src="assets/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="assets/vendor/venobox/venobox.min.js"></script>
  <script src="assets/vendor/owl.carousel/owl.carousel.min.js"></script>
  <script src="assets/vendor/aos/aos.js"></script>

  <!-- Template Main JS File -->
  <script src="assets/js/main.js"></script>


    <!--script src="{{ url_for('static',filename='vendor/jquery/jquery.min.js') }}"></script>
  <script src="{{ url_for('static',filename='vendor/bootstrap/js/bootstrap.bundle.min.js')
}}"></script>
  <script src="{{ url_for('static',filename='vendor/jquery.easing/jquery.easing.min.js')
}}"></script>
  <script src="{{ url_for('static',filename='vendor/php-email-form/validate.js') }}"></script>
  <script src="{{ url_for('static',filename='vendor/waypoints/jquery.waypoints.min.js')
}}"></script>
  <script src="{{ url_for('static',filename='vendor/isotope-layout/isotope.pkgd.min.js')
}}"></script>
  <script src="{{ url_for('static',filename='vendor/owl.carousel/owl.carousel.min.js')
}}"></script>
  <script src="{{ url_for('static',filename='vendor/aos/aos.js') }}"></script>
  <script src="{{ url_for('static',filename='vendor/venobox/venobox.min.js') }}"></script>

  <script src="{{ url_for('static',filename='js/main.js') }}></script-->

</body>

</html>
```

Final.html:

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Prediction</title>
        <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>
```

```
            <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
            <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>
            <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
            <link rel="stylesheet" href="{{ url_for('static', filename='css/final1.css') }}">


<style>
.login{
top: 20%;
}
</style>
</head>

<body>
<div class="header">
<div>CMREC</div>
            <ul>
                        <li><a href="D:/Projects/Major Project/Detection of Phishing
Websites/Flask/index.html">Contact</a></li>
                        <li><a href="D:/Projects/Major Project/Detection of Phishing
Websites/Flask/index.html">About</a></li>
                        <li><a href="D:/Projects/Major Project/Detection of Phishing
Websites/Flask/index.html">Home</a></li>
            </ul>
</div>

<div class="main">
<h1>Phishing Website Detection using Machine Learning</h1>
</div>
<form action="{{ url_for('y_predict')}}"method="post">
            <input type="text" name="URL" placeholder="Enter the URL to be verified"
required="required" />
                        <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
                        </form>
                        <br>
                        <br>

                        <div id='result',class='result' style='color:black;font-size:30px;'>{{
prediction_text }}</div>
                        <a href=" {{ url }} "> {{ url }} </a>
</body>
</html>
```

6.3 Data Dictionary:

1) having_IP_Address { -1,1 }
2) URL_Length { 1,0,-1 }
3) Shortining_Service { 1,-1 }
4) having_At_Symbol { 1,-1 }
5) double_slash_redirecting { -1,1 }
6) Prefix_Suffix { -1,1 }
7) having_Sub_Domain { -1,0,1 }
8) SSLfinal_State { -1,1,0 }
9) Domain_registeration_length { -1,1 }
10) Favicon { 1,-1 }
11) port { 1,-1 }
12) HTTPS_token { -1,1 }
13) Request_URL { 1,-1 }
14) URL_of_Anchor { -1,0,1 }
15) Links_in_tags { 1,-1,0 }
16) SFH { -1,1,0 }
17) Submitting_to_email { -1,1 }
18) Abnormal_URL { -1,1 }
19) Redirect { 0,1 }
20) on_mouseover { 1,-1 }
21) RightClick { 1,-1 }
22) popUpWidnow { 1,-1 }
23) Iframe { 1,-1 }
24) age_of_domain { -1,1 }
25) DNSRecord { -1,1 }
26) web_traffic { -1,0,1 }
27) Page_Rank { -1,1 }
28) Google_Index { 1,-1 }
29) Links_pointing_to_page { 1,0,-1 }
30) Statistical_report { -1,1 }

And finally, the Result designates whether the URL is valid or not:

- Result { -1,1 }

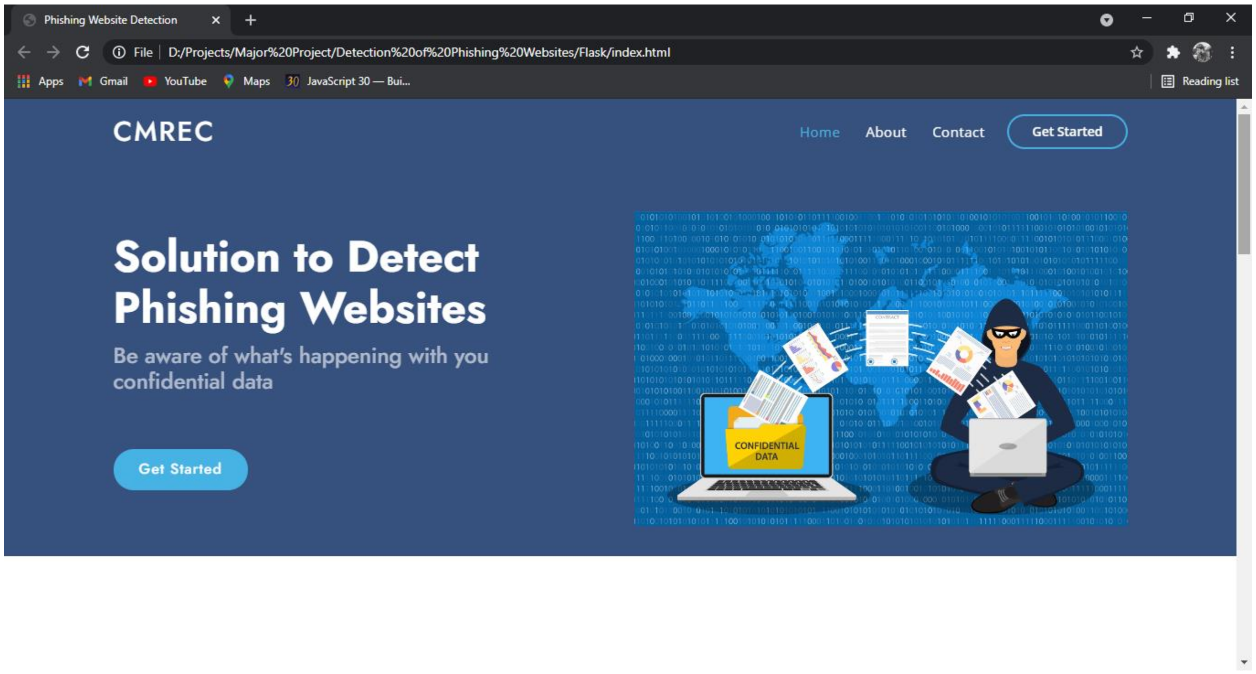
7. SYSTEM TESTING

7.1 Testing Strategies

Manual Testing:

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they use most of the application's features to ensure correct behaviour. To guarantee completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases.

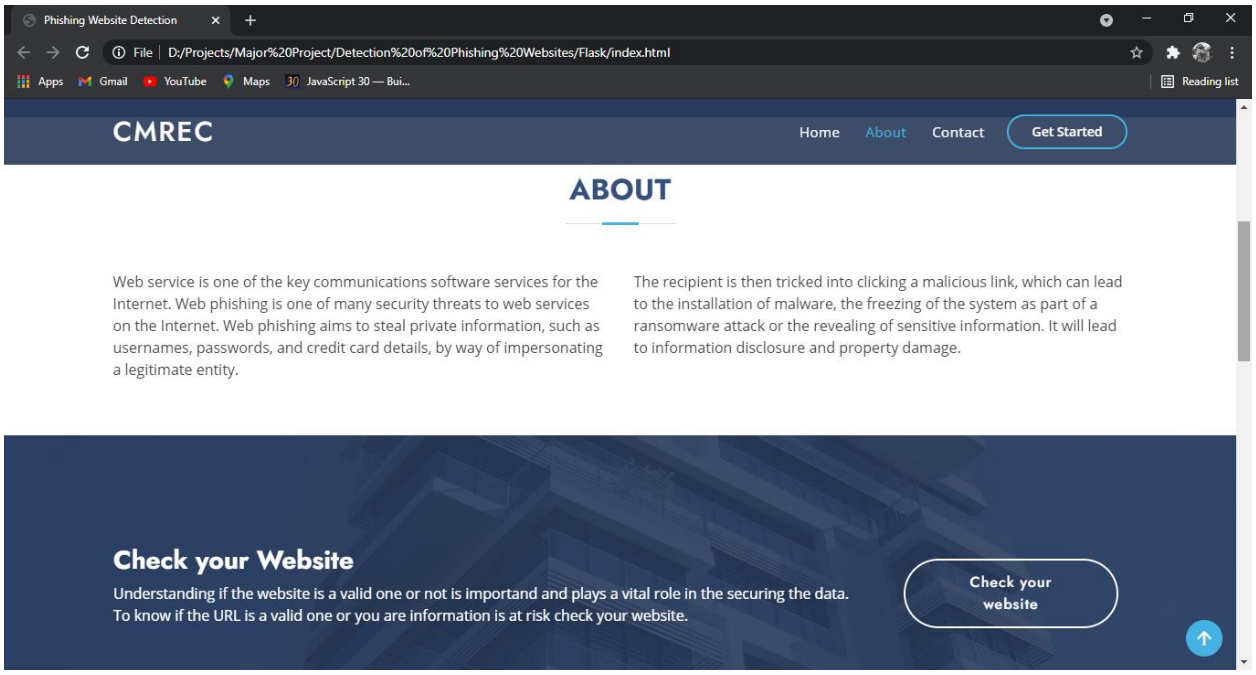
8. OUTPUT SCREENS


Phishing Website Detection
File | D:/Projects/Major%20Project/Detection%20of%20Phishing%20Websites/Flask/index.html
Apps Gmail YouTube Maps JavaScript 30 — Bui...
Reading list
CMREC
Home About Contact Get Started
CONFIDENTIAL DATA

Solution to Detect Phishing Websites

Be aware of what's happening with you confidential data

Get Started


Phishing Website Detection
File | D:/Projects/Major%20Project/Detection%20of%20Phishing%20Websites/Flask/index.html
Apps Gmail YouTube Maps JavaScript 30 — Bui...
Reading list
CMREC
Home About Contact Get Started

ABOUT

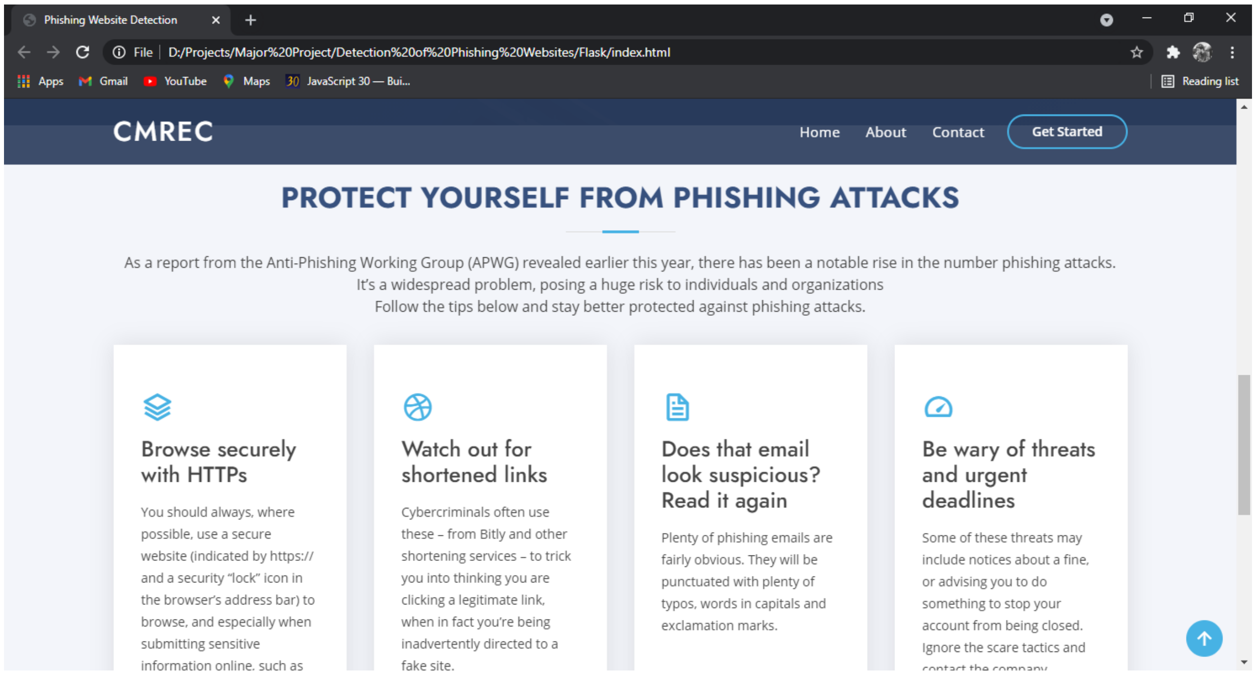
Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet. Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.

The recipient is then tricked into clicking a malicious link, which can lead to the installation of malware, the freezing of the system as part of a ransomware attack or the revealing of sensitive information. It will lead to information disclosure and property damage.

Check your Website

Understanding if the website is a valid one or not is importand and plays a vital role in the securing the data. To know if the URL is a valid one or you are information is at risk check your website.

Check your website



Phishing Website Detection

D:/Projects/Major%20Project/Detection%20of%20Phishing%20Websites/Flask/index.html

CMREC Home About Contact Get Started

PROTECT YOURSELF FROM PHISHING ATTACKS

As a report from the Anti-Phishing Working Group (APWG) revealed earlier this year, there has been a notable rise in the number phishing attacks.
It's a widespread problem, posing a huge risk to individuals and organizations
Follow the tips below and stay better protected against phishing attacks.

Browse securely with HTTPs

You should always, where possible, use a secure website (indicated by https:// and a security "lock" icon in the browser's address bar) to browse, and especially when submitting sensitive information online, such as

Watch out for shortened links

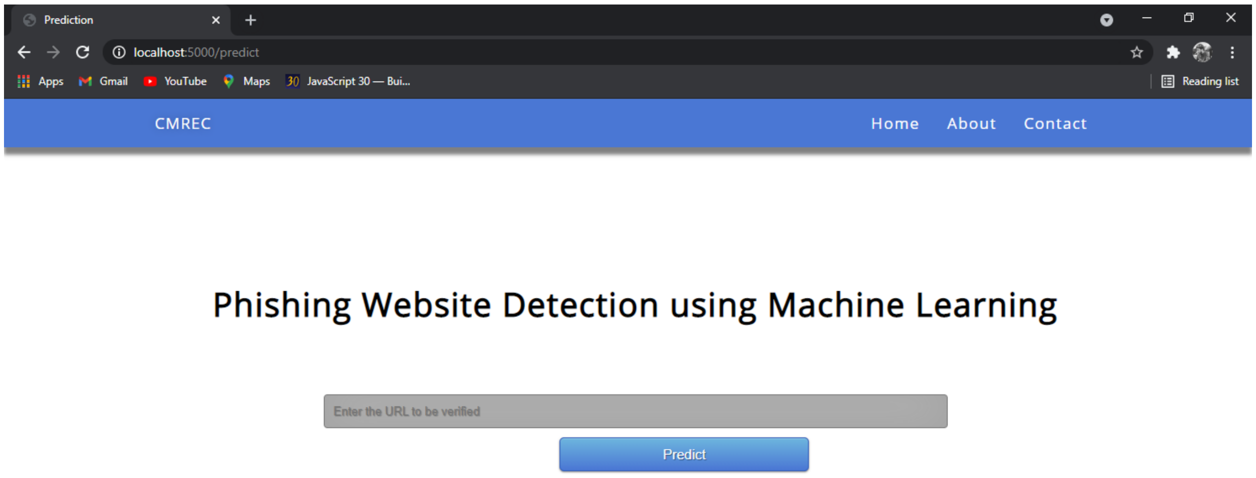
Cybercriminals often use these – from Bitly and other shortening services – to trick you into thinking you are clicking a legitimate link, when in fact you're being inadvertently directed to a fake site.

Does that email look suspicious? Read it again

Plenty of phishing emails are fairly obvious. They will be punctuated with plenty of typos, words in capitals and exclamation marks.

Be wary of threats and urgent deadlines

Some of these threats may include notices about a fine, or advising you to do something to stop your account from being closed. Ignore the scare tactics and contact the company



Prediction

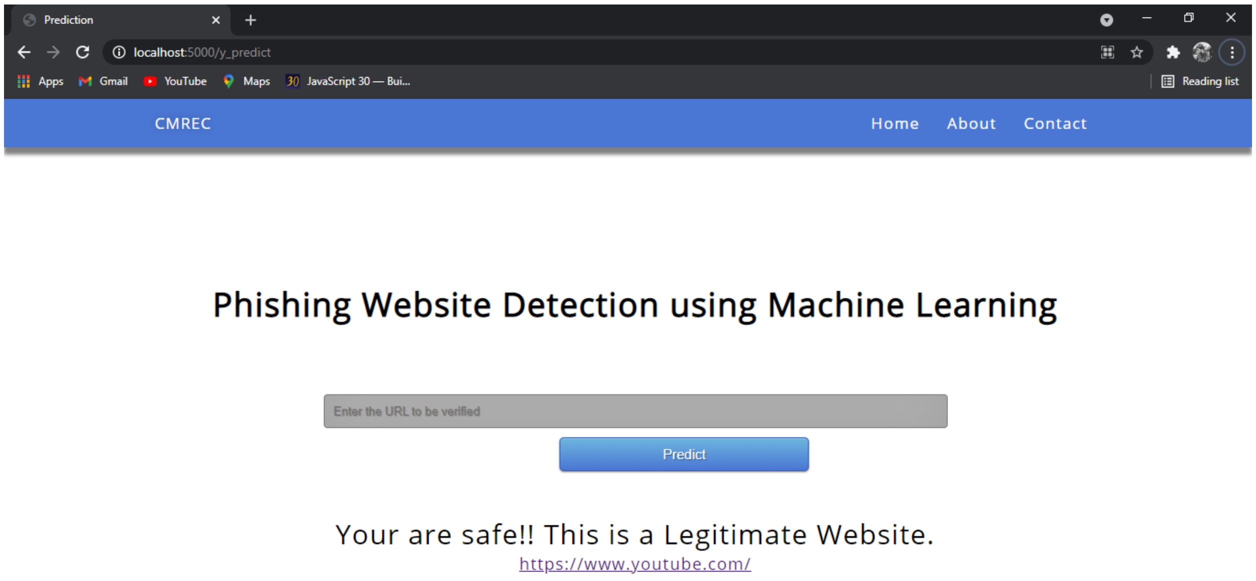
localhost:5000/predict

CMREC Home About Contact

Phishing Website Detection using Machine Learning

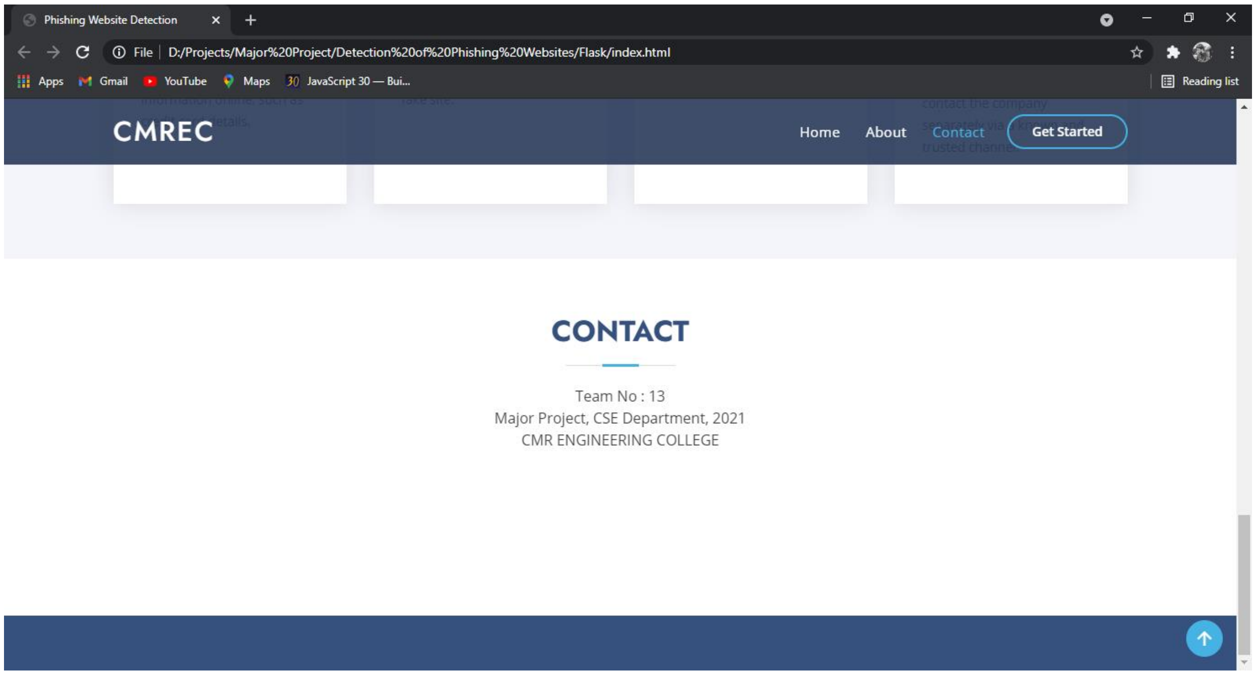
Enter the URL to be verified

Predict



Phishing Website Detection using Machine Learning

Your are safe!! This is a Legitimate Website.
https://www.youtube.com/


Phishing Website Detection
File
D:/Projects/Major%20Project/Detection%20of%20Phishing%20Websites/Flask/index.html
Apps
Gmail
YouTube
Maps
JavaScript 30 — Bui...
Reading list
CMREC
Home
About
Contact
Get Started

CONTACT

Team No : 13
Major Project, CSE Department, 2021
CMR ENGINEERING COLLEGE

9. CONCLUSION

It is found that phishing attacks is very crucial, and it is important for us to get a mechanism to detect it. As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using any of the machine learning algorithm with the classifier. We already have classifiers which gives good prediction rate of the phishing besides, but after our survey that it will be better to use a hybrid approach for the prediction and further improve the accuracy prediction rate of phishing websites. We have seen that existing system gives less accuracy, so we proposed a new phishing method that employs URL based features and also, we generated classifiers through several machine learning. We have got the desired results of testing the site is phishing or not by using four different classifiers. Refer the graph below for the exact results.

10. FUTURE ENHANCEMENTS

In future if we get structured dataset of phishing, we can perform phishing detection much faster than any other technique. In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features, Content based features, Webpage 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.

11. BIBLIOGRAPHY AND REFERENCES

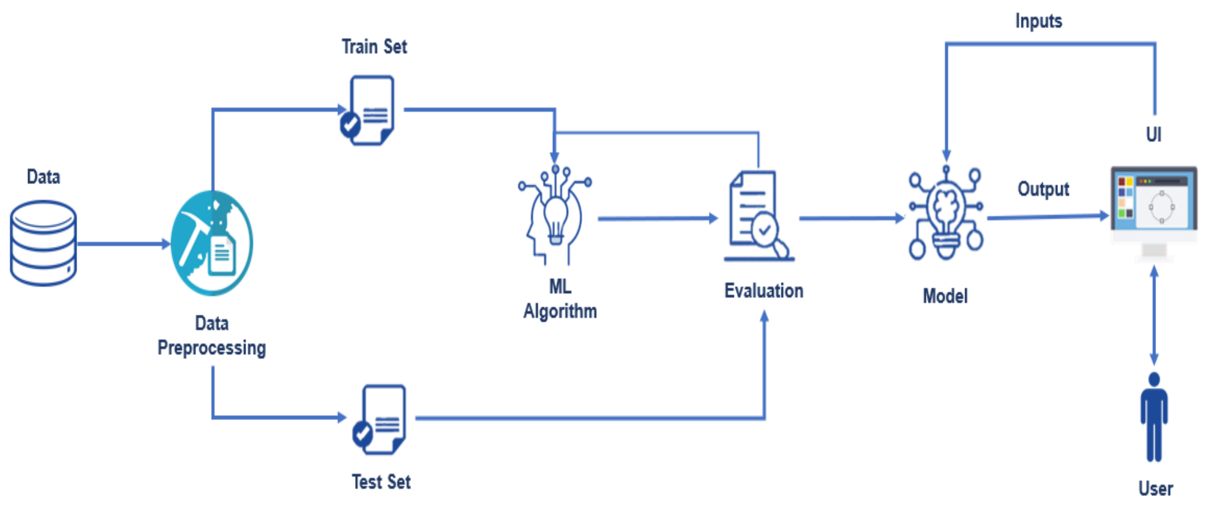
- https://www.sciencedirect.com/science/article/abs/pii/S0957417418306067

- https://www.semanticscholar.org/paper/Machine-learning-based-phishing-detection-from-URLs-Sahingoz-Buber/8dd4a8eefa366b1b7d2471c1b8580df5bea23924

- https://towardsdatascience.com/whataphish-detecting-phishing-websites-e5e1f14ef1a9

12. APPENDICES

12.1 ARCHITECTURE:



12.2 Techniques

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms can be chosen according to the objective. As the dataset which we are using is a Classification dataset so you can use the following algorithms

- Logistic Regression
- Random Forest Regression / Classification
- Decision Tree Regression / Classification
- K-Nearest Neighbors
- Support Vector Machine