Web Phishing Detection

Date	24-11-2022
Team Id	PNT2022TMID39178
Team Leader	Parthasarathi M
Team Member 1	Akash P
Team Member 2	Anil Varma R
Team Member 3	Santhosh H

1. INTRODUCTION

1.1Project Overview

- 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.
- Mainly focuses on applying a machine-learning algorithm to detect Phishing websites.

1.2Purpose

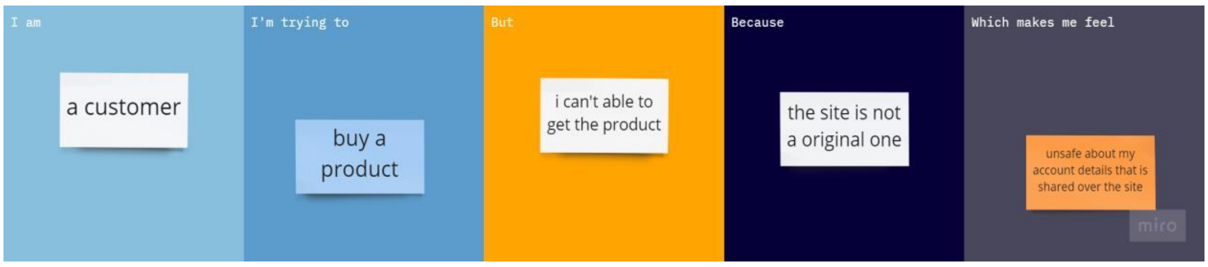
- There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing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

- 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.

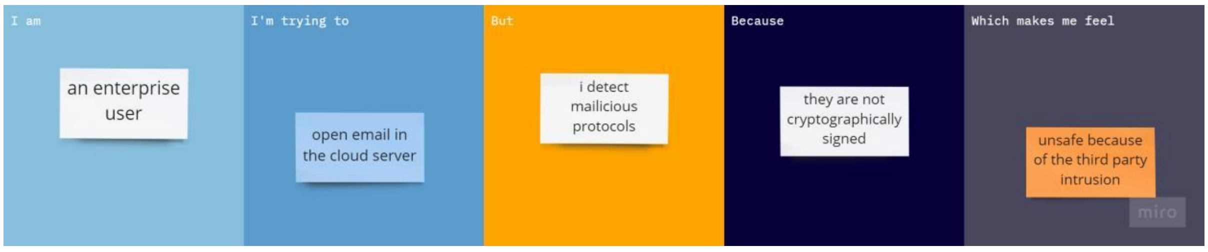
2. Problem Statement Definition

Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. Furthermore, page content inspection has been used by some strategies to overcome the false negative problems and complement the vulnerabilities of the stale lists. Moreover, page content inspection algorithms each have different approach to phishing website detection with varying degrees of accuracy. Therefore, ensemble can be seen to be a better solution as it can combine the similarity in accuracy and different error-detection rate properties in selected algorithms.

Problem Statement



I am	I'm trying to	But	Because	Which makes me feel
a customer	buy a product	i can't able to get the product	the site is not a original one	unsafe about my account details that is shared over the site



I am	I'm trying to	But	Because	Which makes me feel
an enterprise user	open email in the cloud server	i detect mailicious protocols	they are not cryptographically signed	unsafe because of the third party intrusion

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS -1	A customer	Buy a product	I can't get it	The site is not an original one	Unsafe about by account details that is shared over the site
PS -2	An enterprise user	Access email in the cloud server	I detect malicious protocols	They are not cryptographically correct	Unsafe because of the third party intrusion

3. Literature survey

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

Literature Review:

Rao et al. [1] proposed a novel classification approach that use heuristic-based feature extraction approach. In this, they have classified extracted features into three categories suchas URL Obfuscation features, Third-Party-based features, Hyperlink-based features.
Moreover, proposed technique gives 99.55% accuracy. Drawback of this is that as this modeluses third party features, classification of website dependent on speed of third-party services.Also this model is purely depends on the quality and quantity of the training set and Broken links feature extraction has a Volume 3. Chunlin et al. [2] proposed approach that primarily focus on character frequency features. Inthis they have combined statistical analysis of URL with machine learning technique to get result that is more accurate for classification

of malicious URLs. Also they have compared six machine-learning algorithms to verify the effectiveness of proposed algorithm which gives 99.7% precision with false positive rate less than 0.4%.

Sudhanshu et al. [3] used association data mining approach. They have proposed rule based classification technique for phishing website detection. They have concluded that association classification algorithm is better than any other algorithms because of their simple rule transformation. They achieved 92.67% accuracy by extracting 16 features but this is not up to mark so proposed algorithm can be enhanced for efficient detection rate.

M. Amaad et al.[4] presented a hybrid model for classification of phishing website. In this paper, proposed model carried out in two phase. In phase 1,they individually perform classification techniques, and select the best three models based on high accuracy and otherperformance criteria. While in phase 2, they further combined each individual model with best three model and makes hybrid model that gives better accuracy than individual model. They achieved 97.75% accuracy on testing dataset. There is limitation of this model that it requires more time to build hybrid model.

Hossein et al.[5] developed an open-source framework known as “Fresh-Phish”. For phishing websites, machine-learning data can be created using this framework. In this, they have used reduced features set and using python for building query .They build a large labelled dataset and analyse several machine-learning classifiers against this dataset .Analysis of this gives very good accuracy using machine-learning classifiers. These analyses how long time it takesto train the model.

Gupta et al. [6] proposed a novel anti phishing approach that extracts features from client- side only. Proposed approach is fast and reliable as it is not dependent on third party but it extracts features only from URL and source code. In this paper, they have achieved 99.09%of overall detection accuracy for phishing website. This paper have concluded that this approach has limitation as it can detect webpage written in HTML .Non-HTML webpage cannot detect by this approach.

Bhagyashree et al.[7] proposed a feature based approach to classify URLs as phishing and nonphishing. Various features this approach uses are lexical features, WHOIS features, Page Rank and Alexa rank and Phish Tank-based features for disguising phishing and non- phishing website. In this paper, web-mining classification is used.

Mustafa et al.[8] developed safer framework for detecting phishing website. They have extracted URL features of website and using subset based selection technique to obtain betteraccuracy .In this paper, author evaluated CFS subset based and content based subset selectionmethods And Machine learning algorithms are used for classification purpose.

Priyanka et al.[9] proposed novel approach by combining two or more algorithms. In this paper ,author has implemented two algorithm Adaline and Backpropion along with SVM forgetting good detection rate and classification purpose.
Pradeepthi et al.[10] In this paper ,Author studied different classification algorithm and concluded that tree-based classifier are best and gives better accuracy for phishing URL detection. Also Author uses various Volume 3, Issue 7, September-October-2018 | http:// ijsrcseit.com Purvi Pujara et al. Int J S Res CSE & IT. 2018 September-October-2018; 3(7) : 395-399 398 features such as lexical features, URL based feature, network based features anddomain based feature.

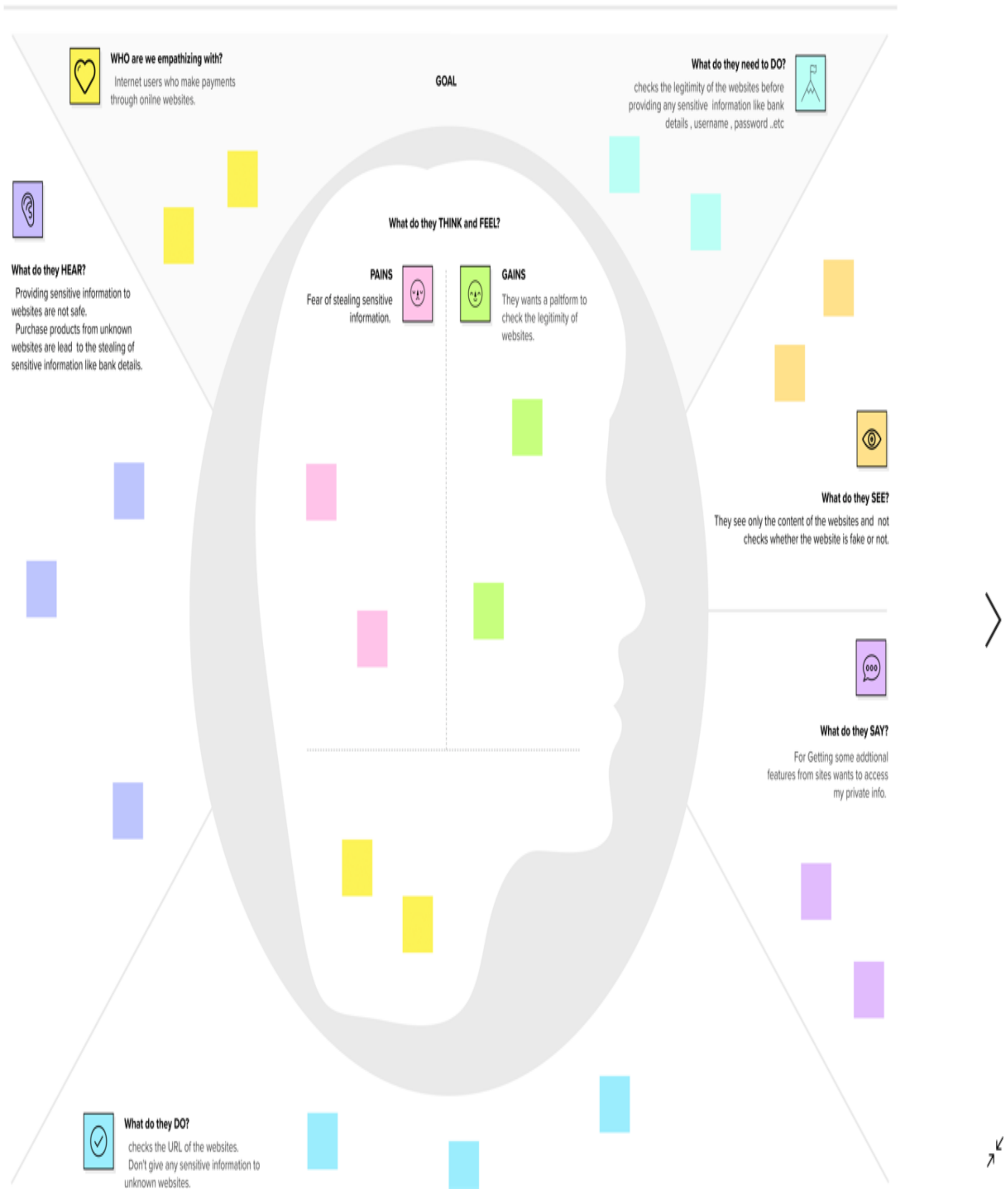
Luong et al. [11] proposed new technique to detect phishing website. In proposed method, Author used six heuristics that are primary domain, sub domain, path domain, page rank, andalexa rank, alexa reputation whose weight and values are evaluated. This approach gives 97
% accuracy but still improvement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

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

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

4. IDEATION & PROPOSED SOLUTION

4.1Empathy Map Canvas



WHO are we empathizing with?
Internet users who make payments through onilne websites.

GOAL

What do they need to DO?
checks the legitimity of the websites before providing any sensitive information like bank details , username , password ..etc

What do they HEAR?
Providing sensitive information to websites are not safe.
Purchase products from unknown websites are lead to the stealing of sensitive information like bank details.

What do they THINK and FEEL?

PAINS
Fear of stealing sensitive information.

GAINS
They wants a paltform to check the legitimity of websites.

What do they SEE?
They see only the content of the websites and not checks whether the website is fake or not.

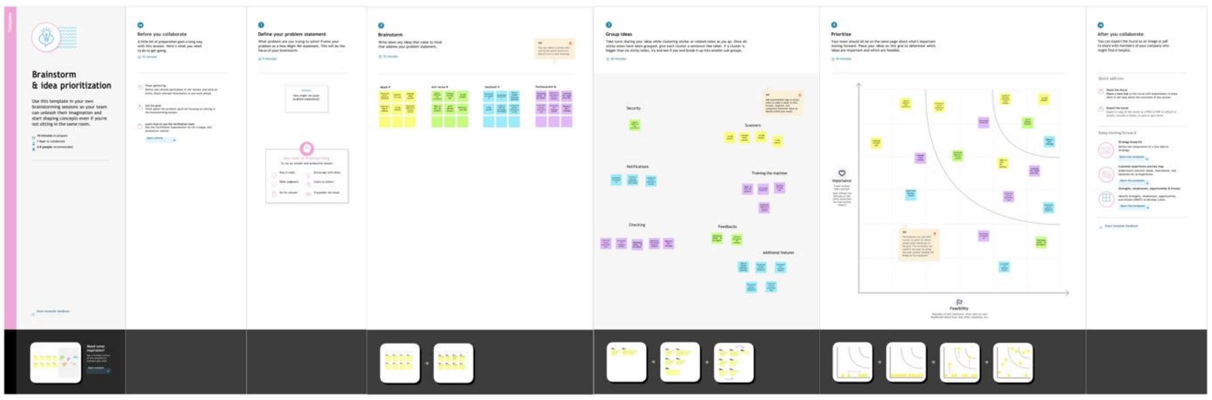
What do they SAY?
For Getting some addtional features from sites wants to access my private info.

What do they DO?
checks the URL of the websites.
Don't give any sensitive information to unknown websites.

4.2Ideation & Brainstorming

Web Phishing Detection using Machine Learning

Today, the Internet covers worldwide. All over the world, people prefer an E-commerce platform to buy or sell their products. Therefore, cybercrime has become the center of attraction for cyber attackers in cyberspace. Phishing is one such technique where the unidentified structure of the Internet has been used by attackers/criminals that intend to deceive users with the use of the illusory website and emails for obtaining their credentials (like account numbers, passwords, and PINs). Consequently, the identification of a phishing or legitimate web page is a challenging issue due to its semantic structure, a phishing detection system is implemented using deep learning techniques to prevent such attacks. The system works on URLs by applying a convolutional neural network (CNN) to detect the phishing webpage, the proposed model has achieved 97.98% accuracy whereas our proposed system achieved accuracy of 98.00% which is better than earlier model. This system doesn't require any feature engineering as the CNN extract features from the URLs automatically through its hidden layers. This is other advantage of the proposed system over earlier reported as the feature engineering is a very time-consuming task.



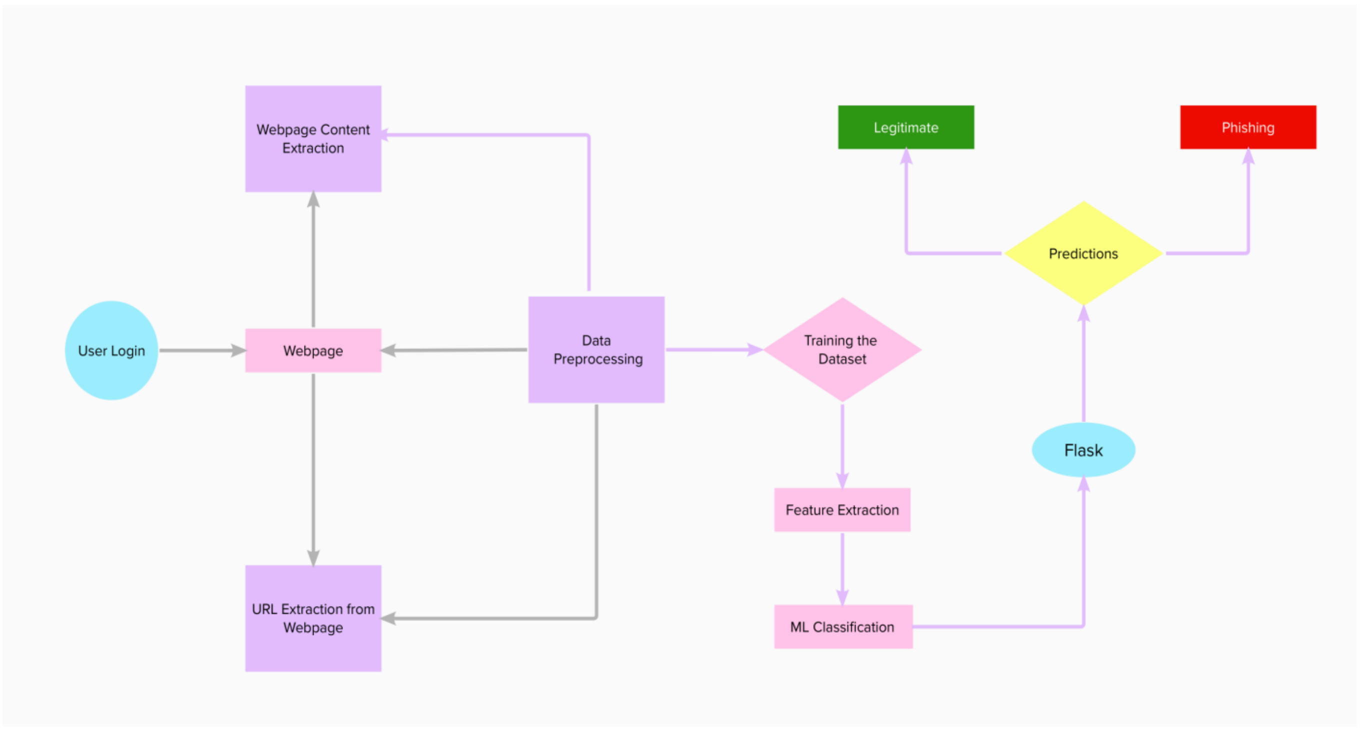
5. REQUIREMENT ANALYSIS

5.1Functional requirement

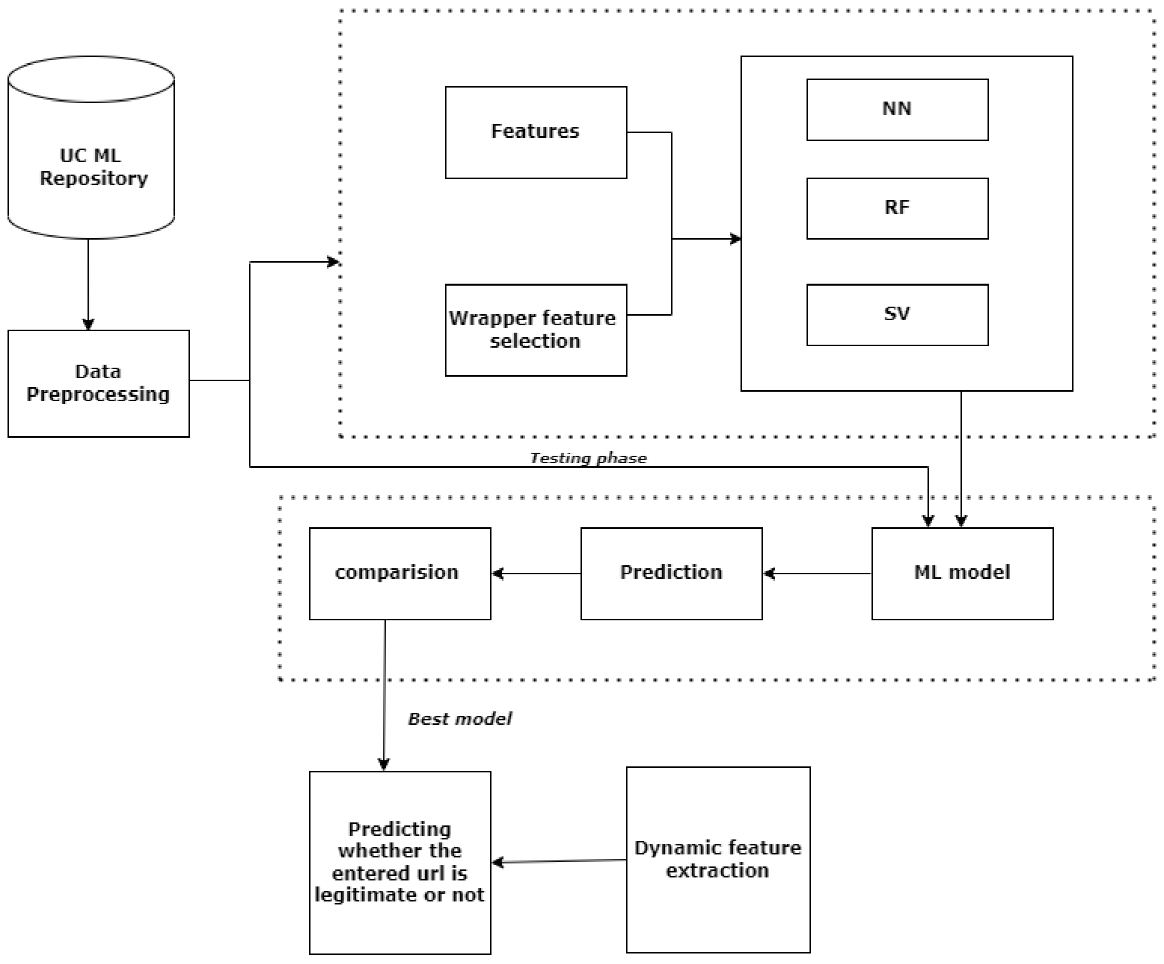
FR NO.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Input	User inputs an URL in required field to checkits validation.
FR-2	Website Comparison	Model compares the websites using Blacklist and Whitelist approach.
FR-3	Feature extraction	After comparing, if none found on comparison then it extracts feature using heuristic and visualsimilarity approach.
FR-4	Prediction	Model predicts the URL using Machine Learning algorithms such as Logistic Regression, KNN,XGBOOST,CNN.
FR-5	Classifier	Model sends all output to classifier and produces final result.
FR-6	Announcement	Model then displays whether website is a legalsite or a phishing site.
FR-7	Events	This model needs the capability of retrievingand displaying accurate result for a website

6. PROJECT DESIGN

6.1Data Flow Diagrams



Architecture



UC ML Repository
Data Preprocessing
Features
Wrapper feature selection
NN
RF
SV
Testing phase
comparision
Prediction
ML model
Best model
Predicting whether the entered url is legitimate or not
Dynamic feature extraction

Fig: Architecture for the web phising detection

Figure description:

NN - Neural Network
RF - Random Forest
SV - Support vector

7. PROJECT PLANNING & SCHEDULING

7.1Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	User input	USN-1	User inputs an URL in the required field to checkits validation.	1	Medium
Sprint-1	Website Comparison	USN-2	Model compares the websites using Blacklistand Whitelist approach.	1	High
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparisonthen it extract feature using heuristic and visual similarity.	2	High
Sprint-2	Prediction	USN-4	Model predicts the URL using Machine learningalgorithms such as logistic Regression, KNN, Tree Regression.	1	Medium
Sprint-3	Classifier	USN-5	Model sends all the output to the classifier andproduces the final result and predict.	1	Medium
Sprint-4	Announcement	USN-6	Model then displays whether the website is legal site or a phishing site.	1	High
Sprint-4	Events	USN-7	This model needs the capability of retrieving anddisplaying accurate result for a website.	1	High

Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed(as on planned end date)	Sprint End Date(Actual)
Sprint-1	20	6 days	24 October 2022	12 November 2022	20	29 October 2022
Sprint-2	20	6 days	31 October 2022	14 November 2022	20	05 November 2022
Sprint-3	20	6 days	07 November 2022	16 November 2022	20	12 November 2022
Sprint-4	20	6 days	14 November 2022	19 November 2022	20	19 November 2022

8. CODING & SOLUTIONING (Explain the features added in the project along with code)

8.1Feature 1

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <!-- BootStrap -->
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
       integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1d
KGj7Sk" crossorigin="anonymous">

   <link href="static/styles.css" rel="stylesheet">
```

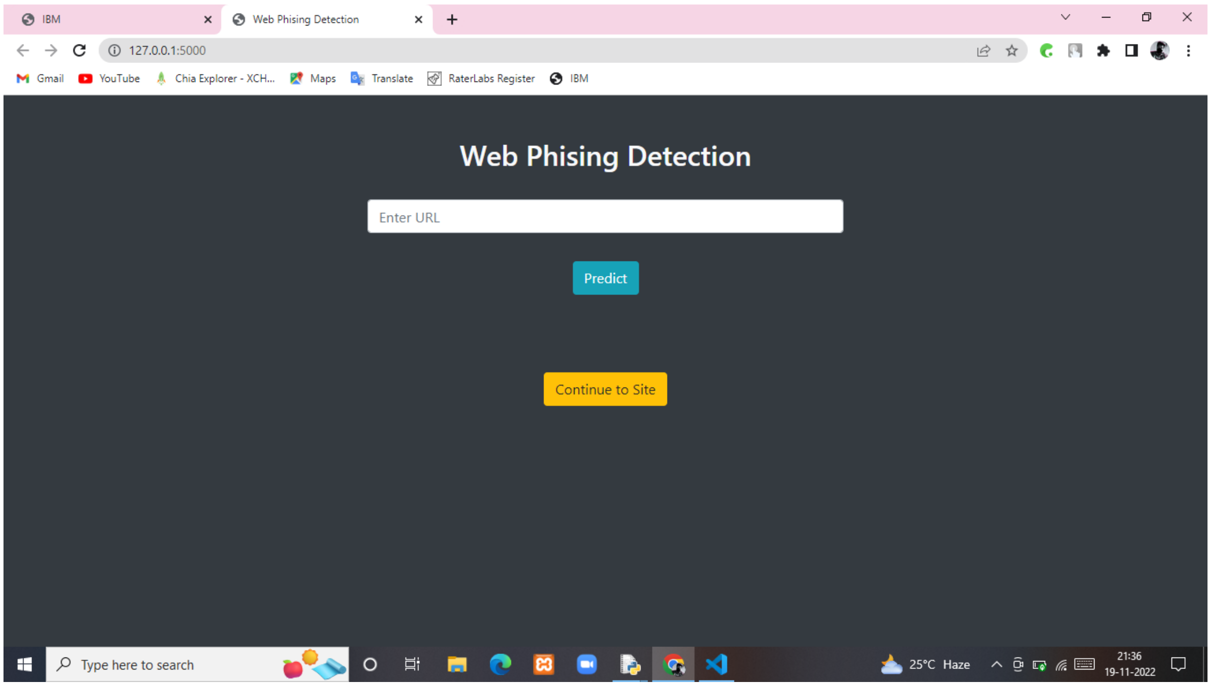
```html
    <title>IBM Nalaiya Thiran</title>
</head>

<body class="bg-dark">
<div class="container mt-5">
    <div>
        <center>
        <div class="form col-md text-light" id="form1">
            <center>
            <h2>Web Phishing Detection using Machine Learning</h2>
            <br>
            <form action="/" method ="post" autocomplete="off">
                <input type="text" class="form-control w-50" name ='url' id="url"
placeholder="Enter URL" required="" />
                <br>
                <button class="btn btn-info mt-2" role="button" >Predict</button>
            </form>
        </div>
        <br>
        <div class="col-md" id="form2">
            <br>
            <h4 class = "right "><a href= {{ url }} target="_blank">{{ url
}}</a></h4>
            <br>
            <h3 id="prediction" class="text-warning"></h3>
            <button class="btn btn-warning" id="btn1"
role="button"  onclick="window.open('{{url}}')" target="_blank">Continue to
Site</button>
        </div>
        </center>
    </div>
    <br>
</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCX
aRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao"
        crossorigin="anonymous"></script>
```

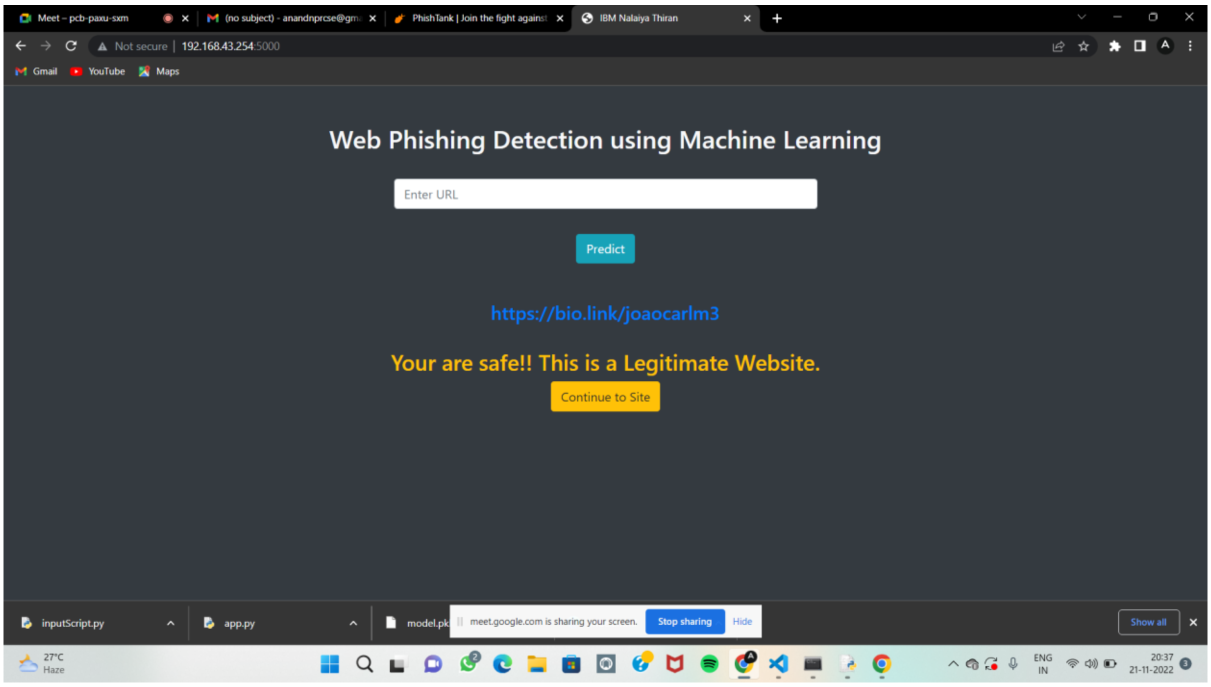
```
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
    integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR
0JKI"
    crossorigin="anonymous"></script>


  <script defer>
    document.querySelector("#btn1").style.display = "none";
    let result = '{{result}}';
    if(result!==undefined || result!==null){
      console.log(result)
      document.getElementById("prediction").innerHTML = result;
      document.getElementById("btn1").style.display="inline-block";
    }
  </script>

</body>
</html>
```

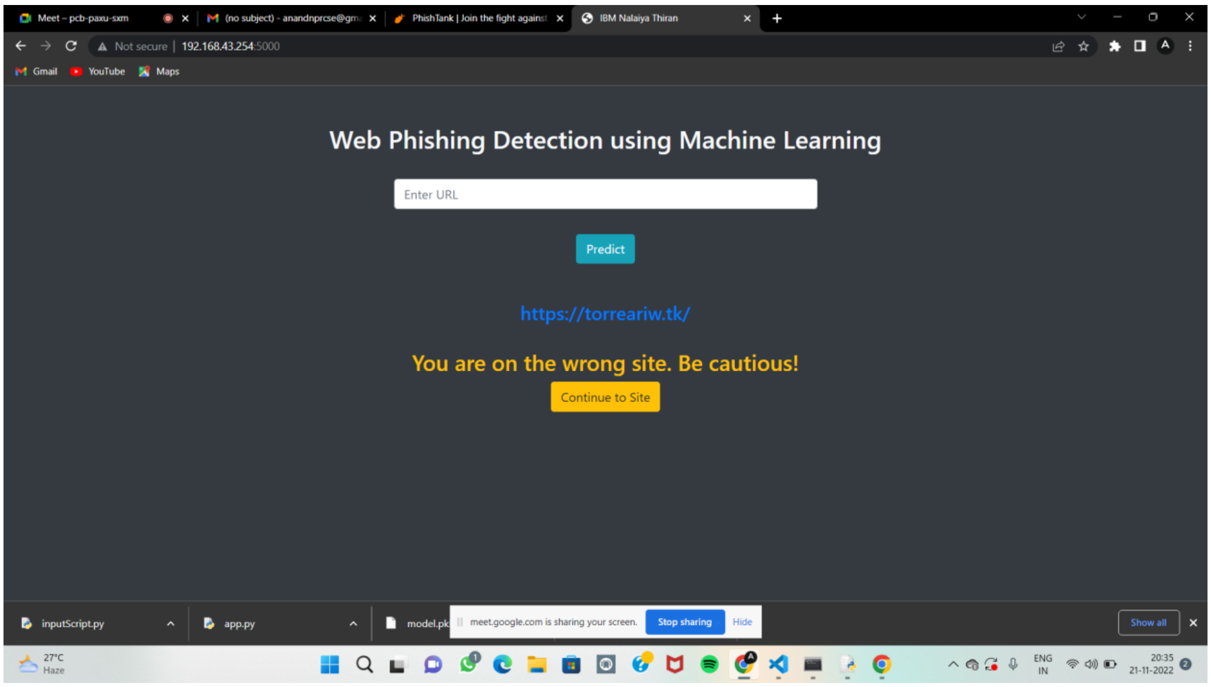
- We Build an HTML page to take the URL as a text and upon clicking on the button for submission it has to redirect to the URL for “y_predict” which returns if the URL given is phishing or safe. The output is to be then displayed on the page.

8.2Feature 2 (Detection Result)

When the URL is given, the model analyses and gives the output whether it is a phishing or legitimate website.


IBM
Web Phising Detection
127.0.0.1:5000
Gmail
YouTube
Chia Explorer - XCH...
Maps
Translate
RaterLabs Register
IBM
Web Phising Detection
Enter URL
Predict
Continue to Site
Type here to search
25°C Haze
21:36
19-11-2022


Meet – pcb-paxu-sxm
(no subject) - anandnprcse@gm
PhishTank | Join the fight against
IBM Nalaiya Thiran
Not secure | 192.168.43.254:5000
Gmail
YouTube
Maps
Web Phishing Detection using Machine Learning
Enter URL
Predict
https://bio.link/joaocarlm3
Your are safe!! This is a Legitimate Website.
Continue to Site
inputScript.py
app.py
model.pk
meet.google.com is sharing your screen.
Stop sharing
Hide
Show all
27°C
Haze
ENG
IN
20:37
21-11-2022


Web Phishing Detection using Machine Learning
Enter URL
Predict
https://torreariw.tk/
You are on the wrong site. Be cautious!
Continue to Site

9. Testing

User Acceptance Testing

Acceptance Testing
UAT Execution & Report Submission

1. Purpose of Document

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

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

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done. The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

2. Test Case Analysis

This reports how the number of test cases that have passed, failed, and untested

UAT is performed by –

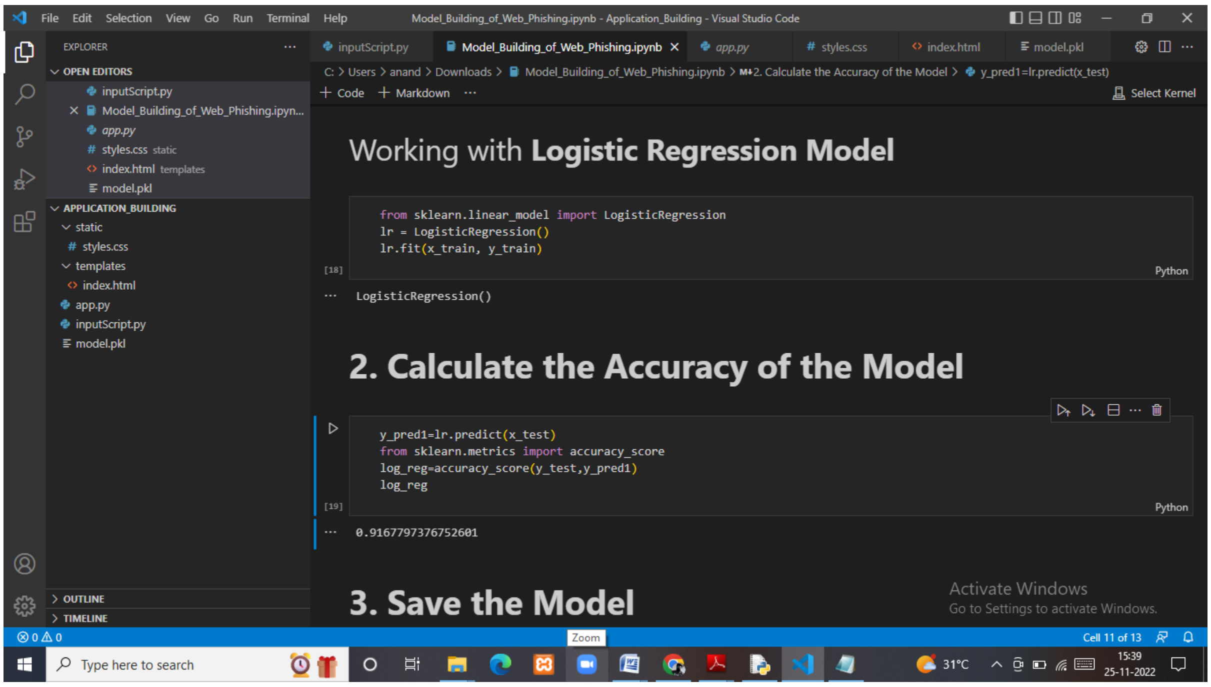
· Client

· End users

Phishing detection is done using 16 different heuristic rules. In the system, 11 main classes were defined, and 1 class was defined with 5 sub-classes. This covers all 16 heuristic rules. To test the system, 15 test cases were designed using assertion methods. Ten test cases were designed to test the 10 main classes and 5 test cases were designed to test the class with five sub-classes. The getter-setter method was used to test the class with five sub-classes. The getter method is used to obtain or retrieve a variable value from the class, and the setter method is used to store the variables. The class with five sub-classes checks the 5 different heuristic rules, length of the URL, number of dots and slashes in the URL, presence of @ symbols in the URL, IP address mentioned in the URL, and the presence of special character such as ',', '_', ';' in the URL. Initially, only a single test case was created for the class with five sub-classes, but it was failing as this class has five methods. After applying the getter setter method, all the test cases passed without any issues. The test results are shown in assert Not Null() is used to check if the input URL is not empty, and assert Array Equals() is used to compare the result from the detection method with the expected result.

10.RESULTS

Performance Metrics


Working with **Logistic Regression Model**

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

```
LogisticRegression()
```

2. Calculate the Accuracy of the Model

```
y_pred1=lr.predict(x_test)
from sklearn.metrics import accuracy_score
log_reg=accuracy_score(y_test,y_pred1)
log_reg
```

```
0.9167797376752601
```

3. Save the Model

11.ADVANTAGES

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

DISADVANTAGES

- If Internet connection fails, this system won’t work.
- All websites related data will be stored in one place.

12.CONCLUSION

- Due to the growing use of Internet in our daily life, cyber attackers aim their victim over this platform. One of the mostly encountered attack is named as "phishing" which creates as poofed web page to obtain the users sensitive information such as user-ID and password in financial websites by using social networking facilities. The malicious web page is created as if a legitimate web page, especially copying the original web page one to one. Therefore, detection of these pages is a very trivial problem to overcome due to its semantic structure which takes the advantage of the humans' vulnerabilities Software tools can only be used as a support mechanism for detection and prevention this type attacks, and these tools especially use whitelist/blacklist approach to overcome this type of attacks. However, they are static algorithms and cannot identify the new type of attacks in the system. Therefore, as an efficient solution, we propose the use of logistic regression machine learning system for classifying the incoming URLs. The experimental results show that this approach result satisfactory accuracy rate of about 97% of accuracy.

13.FUTURE SCOPE

- As the Future works, to decrease the execution time and increase the efficiency of the system,the power of the Graphics Programming Units can be used. Additionally, other approaches of Deep Learning, such as recurrent neural networks and convolutional neural networks can be tested for increasing the performance of the system.

APPENDIX

Source Code

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1d
KGj7Sk" crossorigin="anonymous">

    <link href="static/styles.css" rel="stylesheet">
    <title>IBM Nalaiya Thiran</title>
</head>

<body class="bg-dark">
<div class="container mt-5">
    <div>
        <center>
        <div class="form col-md text-light" id="form1">
            <center>
            <h2>Web Phishing Detection using Machine Learning</h2>
            <br>
            <form action="/" method ="post" autocomplete="off">
                <input type="text" class="form-control w-50" name ='url' id="url"
placeholder="Enter URL" required="" />
                <br>
                <button class="btn btn-info mt-2" role="button" >Predict</button>
            </form>
        </div>
        <br>
        <div class="col-md" id="form2">
            <br>
            <h4 class = "right "><a href= {{ url }} target="_blank">{{ url
}}</a></h4>
            <br>
            <h3 id="prediction" class="text-warning"></h3>
            <button class="btn btn-warning" id="btn1"
role="button"  onclick="window.open('{{url}}')" target="_blank">Continue to
Site</button>
        </div>
        </center>
    </div>
    <br>
</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCX
aRkfj"
        crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfoo
Ao"
        crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR
0JKI"
        crossorigin="anonymous"></script>


    <script defer>
        document.querySelector("#btn1").style.display = "none";
        let result = '{{result}}';
        if(result!==undefined || result!==null){
            console.log(result)
            document.getElementById("prediction").innerHTML = result;
            document.getElementById("btn1").style.display="inline-block";
        }
    </script>

</body>
</html>
```

App.py

```
#importing required libraries

import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

import inputScript

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

#Redirects to the page to give the user input URL.
@app.route('/')
def predict():
    return render_template('index.html',result="")

#Fetches the URL given by the URL and passes to inputScript
@app.route('/',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    url = request.form['url']
    checkprediction = inputScript.main(url)
    print(url)
    print(checkprediction)
    prediction = model.predict(X=checkprediction)
    print(prediction)
    output=prediction[0]
    print(output)
    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('index.html', result=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.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)
if __name__ == "__main__":
    app.run(host='0.0.0.0', debug=True)
```

inputScript.py

```
import regex
from tldextract import extract
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import requests
import favicon
import re
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\.com|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\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.5
0\.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\.14
1|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\.23
1\.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(u
rl),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),Ifra
me(url),
            age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),G
oogle_Index(url),
            Links_pointing_to_page(url),Statistical_report(url)]]


    return check
```

REFERENCES

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

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

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

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

[5] Mustafa AYDIN, Nazife BAYKAL : Feature Extraction and Classification PhishingWebsites Based on URL : IEEE,2015

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

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

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

[9] Ahmad Abunadi, Anazida Zainal ,Oluwatobi Akanb: Feature Extraction Process: APhishing Detection Approach :In IEEE,2013.

[10] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey: An Assessment of Features Related to Phishing Websites using an Automated Technique:In The 7th International Conference for Internet Technology and Secured Transactions,IEEE,2012

Github:

https://github.com/IBM-EPBL/IBM-Project-19021-1659692213

Demo Link:

https://drive.google.com/drive/u/0/folders/1fv9X6xX9dqmLeFZH-GwNeL_UNdX9TNkH