**Project Report**

|  |  |
| --- | --- |
| **Date** | 20 November 2022 |
| **Team ID** | PNT2022TMID48697 |
| **Project Name** | Project - Web Phishing Detection |
| **Team Lead** | Prasanna D |
| **Team Member 1** | Thambi Durai P K |
| **Team Member 2** | Mohan S |
| **Team Member 3** | Mohan Babu B |

1. **INTRODUCTION**
   1. Project 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. Purpose
* 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.

1. **LITERATURE SURVEY**
   1. Existing problem

* Cyber criminals use phishing emails because it's easy, cheap and effective. Email addresses are easy to obtain, and emails are virtually free to send. With little effort and cost, **attackers can quickly gain access to valuable data**.
  1. References
* Gunter Ollmann, “The Phishing Guide Understanding &Preventing Phishing Attacks”, IBMInternet Security Systems”, 2007.
* https://resources.infosecinstitute.com/category/enterprise/phishing/the-phishing-landscape/phishing-data-attack-Statistics/#href
* Mahmoud Khonji, Youssef Iraqi, "Phishing Detection: A Literature Survey IEEE, and Andrew Jones, 2013
* www.phishtank.com
  1. 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

.



Designed by: Prasanna D, Thambi Durai P.K

1. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas

1 Who are you empathizing with?

* Who is student you’re trying to understand?

We are Prasanna & Thambidurai and we are trying to detect web phishing.

* What is their situation?

The URL of phishing websites may be very similar to real websites to the human eye, but they are different in IP. **The content-based detection usually refers to the detection of phishing sites through the pages of elements, such as form information, field names, and resource reference**

6 What do they hear?

* What are they hearing others say? Hacker Attacks.
* What are they hearing from their friends? Someone hacked you and

stealed your information.

* What are they hearing from teachers? Data Breach.
* What are they hearing from their parents? Your personal information

belongs to someone we don’t know.

2

Date: 30.10.2022

Aim

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. We need to detect the before attack and solve all conflicts.

What do they need to do?

* What do they need to do differently?

Software-based phishing detection techniques are preferred for fighting against the phishing attack. Mostly available methods for detecting phishing attacks are **blacklists/whitelists, natural language processing, visual similarity, rules, machine learning techniques.**

* What things do they need to get done?

Current methods for phishing detection include **black and whitelists, heuristics, visual similarity, and machine learning**, among which heuristics and machine learning are more widely used.

* How will we know they were successful at this?

Detection Process. Detecting Phishing Domains is a classification problem, so it means we need labeled data which has samples as phish domains and legitimate domains in the training phase. **The dataset which will be used in the training phase** is a very important point to build successful detection mechanism.

7

**What do they think and feel?**

Phishing is the ultimate social engineering attack, giving a hacker the scale and ability to go after hundreds or even thousands of users all at once. **Phishing scams involve sending out emails or texts disguised as legitimate sources**.

4 What do they say?

It occurs when an attacker, masquerading as a trusted entity, dupes a

victim into opening an email, instant message, or text message.

What do they do?

* What behavior have we observed?

Phishing **tricks victims into giving over credentials for all sorts of sensitive accounts**, such as email, corporate intranets and more. Even for cautious users, it's sometimes difficult to detect a phishing attack.

* What can we imagine them doing?

Lack of security awareness among employees is also one of the major reasons for the success of phishing. Organizations should be aware of how the benefits and purpose of security awareness training can secure their employees from falling victim to phishing attacks.

*We acknowledge the original work and source material for empathy maps to be Prasanna and his Teams. ’Web Phishing Detection’ this work with respect and does not claim to have created the original concept.*

* What have you heard them say?
* What can you imagine the sayingd
  1. Ideation & 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.
  1. Proposed Solution

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | Web phishing tends to steal a lots of information from the user during online transaction like username, password, important documents that has been attached to that websites. There are Multiple Types of Attacks happens here every day, but there is no auto detection Process through Machine Learning is achieved |
| 2. | Idea / Solution description | 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, This system doesn’t require any feature engineering as the CNN extract features from the URLs automatically through its hidden layers. |
| 3. | Novelty / Uniqueness | This project not only able to identify the malicious websites it also has the ability to automatically block these kind of websites completely in the future when it has been identified and also blocks some various mails  /ads from these malicious websites |
| 4. | Social Impact / Customer Satisfaction | This web phishing detection project attains the customer satisfaction by discarding various kinds of malicious websites to protect their privacy. This project is not only capable of using by an single individual ,a large social community and a organisation can use this web phishing detection to protect their privacy. This project helps to block various malicious websites simultaneously. |
| 5. | Business Model (Revenue Model) | This developed model can be used as an enterprise applications by organisations which handles sensitive information and also can be sold to government agencies to prevent the loss  of potential important data. |
| 6. | Scalability of the Solution | This project’s performance rate will be high approximately 97% and it also provide many capabilities to the user without reducing its efficieny to detect the  malicious websites as well as industrial limits websites beyond the imitations to find its legitimate. |

* 1. Problem Solution fit

**1. CUSTOMER SEGMENT(S)**

**6. CUSTOMER CONSTRAINTS**

**CS**

**CC**

An enterprise user surfing through the internet for some information.

They don’t know what to do after losing

data.

**5. AVAILABLE SOLUTIONS**

Which solutions are available

The already available solutions are blocking such phishing sites and by triggering a message to the customer about dangerous nature of the website.

**AS**

An internet user who is willing to shop products online.

Customers have very little awareness on phishing websites.

But the blocking of phishing sites are not more affective as the attackers use a different/new site to steal potential data thus a AI/ML model can be used to prevent customers from these kinds of sites from stealing data

5

The option to check the legitimacy of the Websites is provided.

Users get an idea what to do and more importantly what not to do.

**BE**

**7. BEHAVIOUR**

Very limited research is performed on this part of the internet.

**RC**

**9. PROBLEM ROOT CAUSE**

The hackers use new ways to cheat the naive users.

The phishing websites must be detected in a earlier stage .

The user can be blocked from entering such sites for the prevention of such issues.

**J&P**

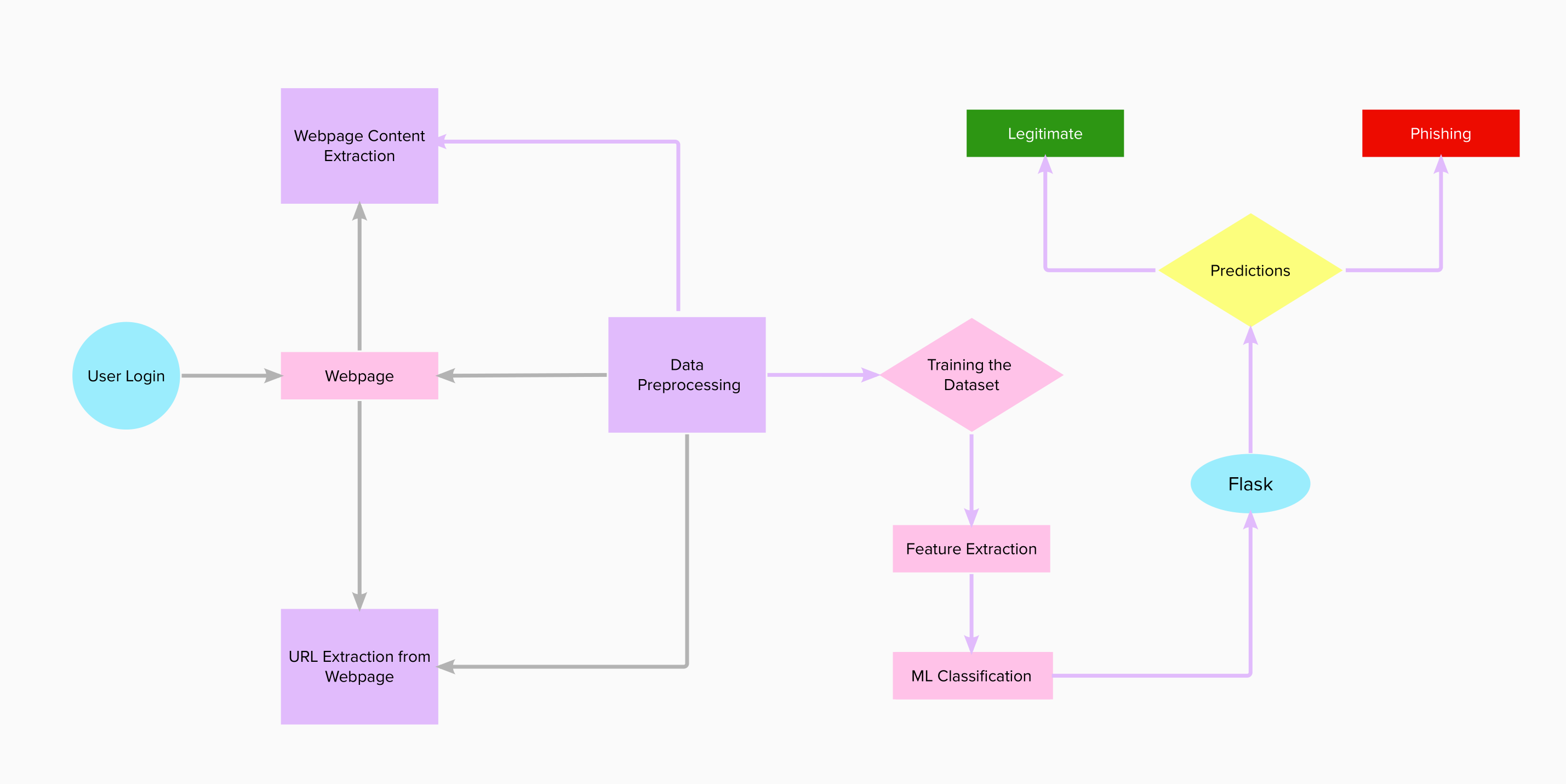
**2. JOBS-TO-BE-DONE / PROBLEMS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I**  **d e n t i f y s t r o n g T R**  **& E M** | **3. TRIGGERS**  A trigger message can be popped warning the user about the site.  Phishing sites can be blocked by the ISP and can show a “site is blocked” or “phishing site detected” message. | **TR** | . | **10. YOUR SOLUTION SL**  An option for the users to check the legitimacy of the websites is provided.  This increases the awareness among users and prevents misuse of data, data theft etc.,  Creating the URL detection site to find the url is phishing or legitimate. | | | **8. CHANNELS of BEHAVIOUR CH**  **8.1 ONLINE**  Customers tend to lose their data to phishing sites.  **8.2 OFFLINE**  Customers try to learn about the ways they get cheated from various resources viz., books, other people etc., | | **I**  **d e n ti f y s t r o n g T R**  **& E M** |
| **4. EMOTIONS: BEFORE / AFTER EM** | | |  |  |  | |  | |
| How do customers feel when they face a problem or a job and afterwards? | | |
| The customers feel lost and insecure to use the internet after facing such issues. | | |
| Unwanted panicking of the customers is felt after encounter loss of potential data to such sites. | | |

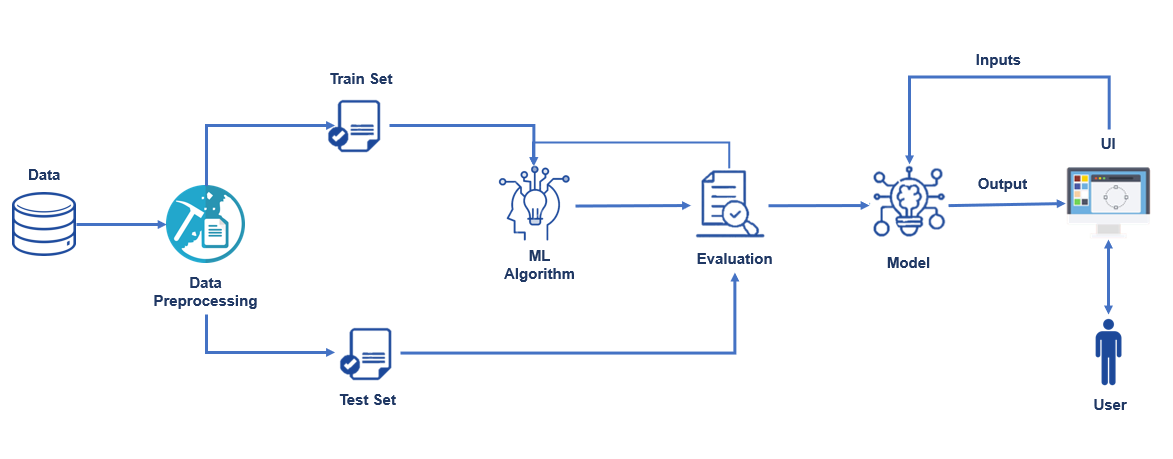
1. **REQUIREMENT ANALYSIS**
   1. Functional requirement

|  |  |  |
| --- | --- | --- |
| **FR NO.** | **Functional**  **Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Input | User inputs an URL in required field to check its 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 visual similarity 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 legal site or a phishing site. |
| FR-7 | Events | This model needs the capability of retrieving and displaying accurate result for a website |

1. **PROJECT DESIGN**
   1. Data Flow Diagrams



* 1. Solution & Technical Architecture



* 1. User Stories

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

1. **PROJECT PLANNING & SCHEDULING**
   1. Sprint 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 check its validation. | 1 | Medium |
| Sprint-1 | Website Comparison | USN-2 | Model compares the websites using Blacklist and Whitelist approach. | 1 | High |
| Sprint-2 | Feature Extraction | USN-3 | After comparison, if none found on comparison then it extract feature using heuristic and visual  similarity. | 2 | High |
| Sprint-2 | Prediction | USN-4 | Model predicts the URL using Machine learning algorithms such as logistic Regression, KNN, Tree Regression. | 1 | Medium |
| Sprint-3 | Classifier | USN-5 | Model sends all the output to the classifier and produces 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 and displaying accurate result for a website. | 1 | High |

* 1. 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 |

1. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   1. Feature 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+NcPb1dKGj7Sk" 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+OrCXaRkfj"

        crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"

        crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"

        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"

        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";

        }

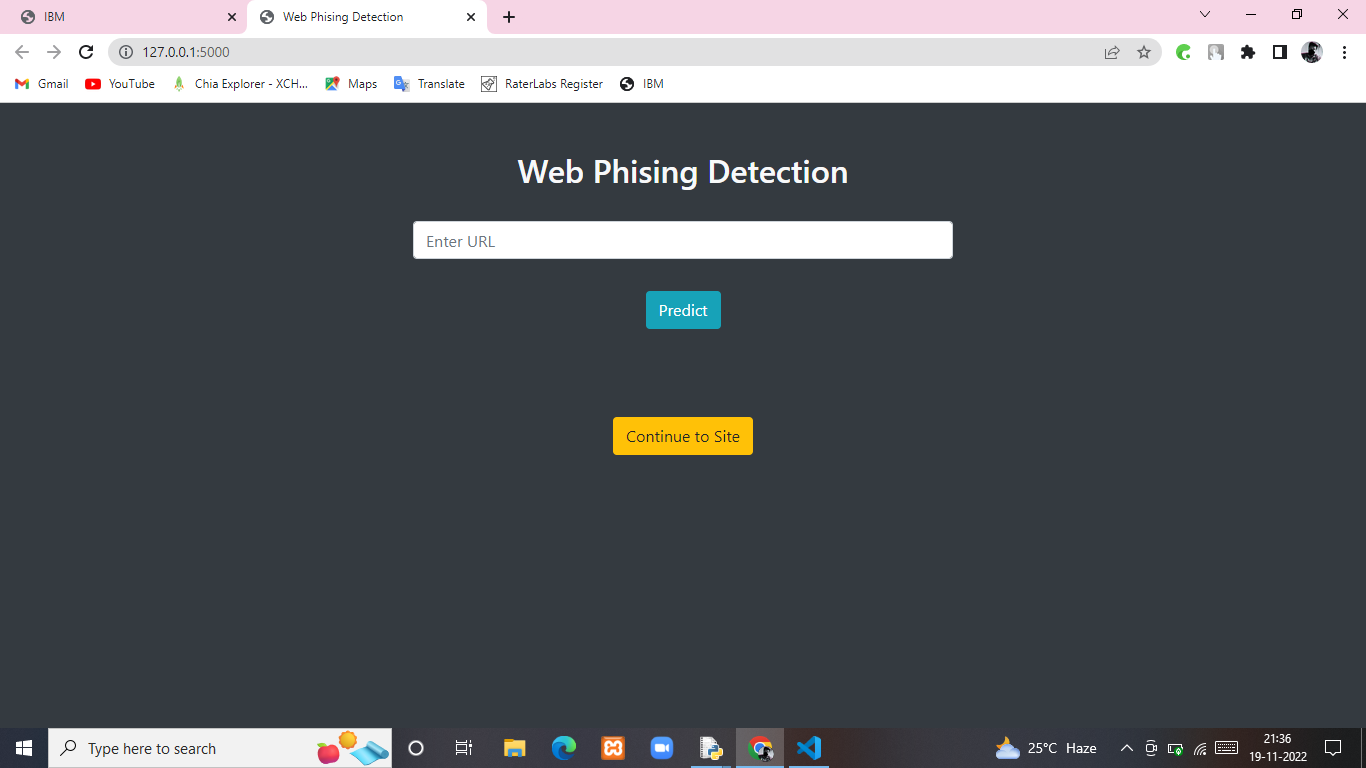
    </script>

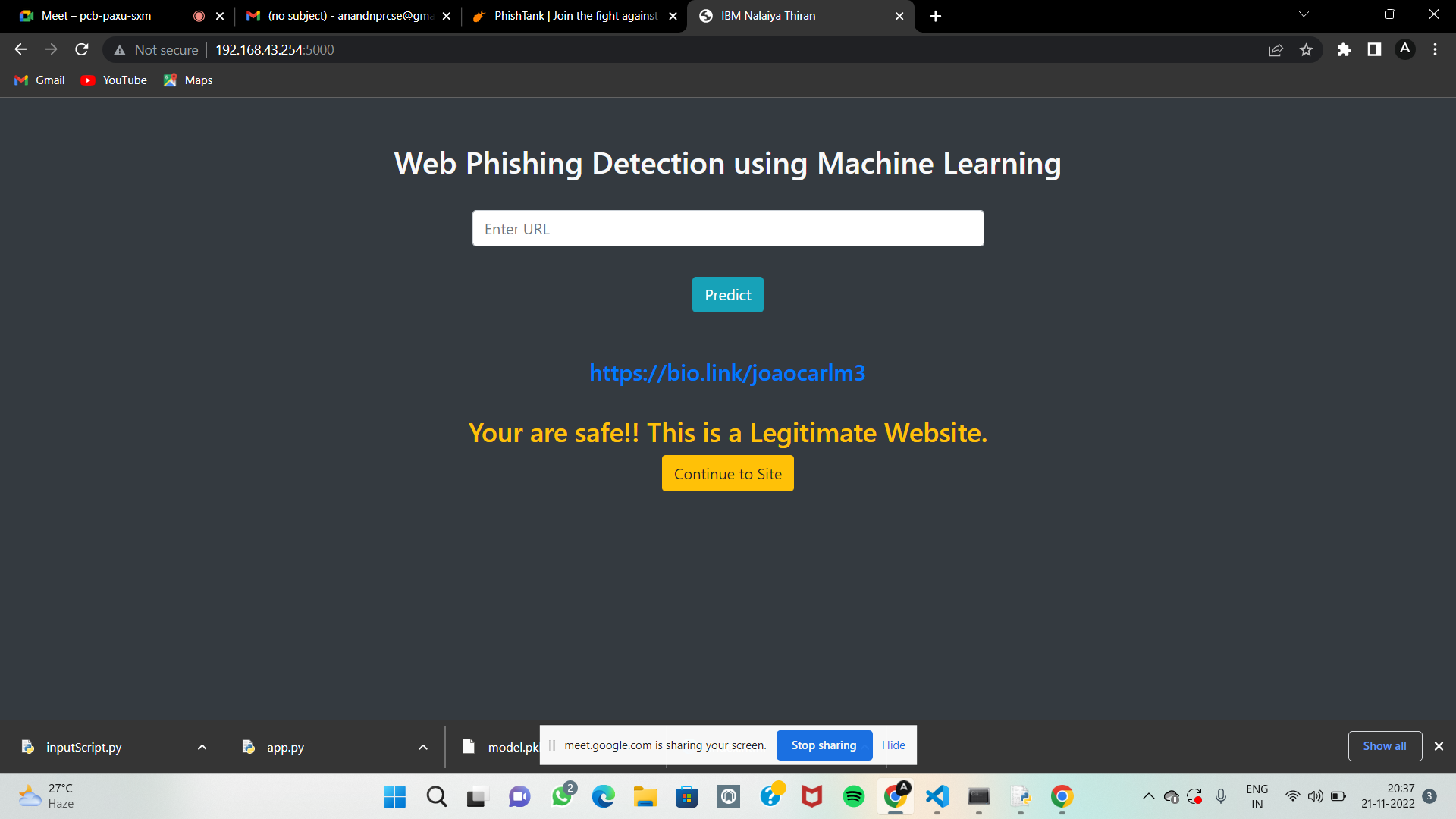
</body>

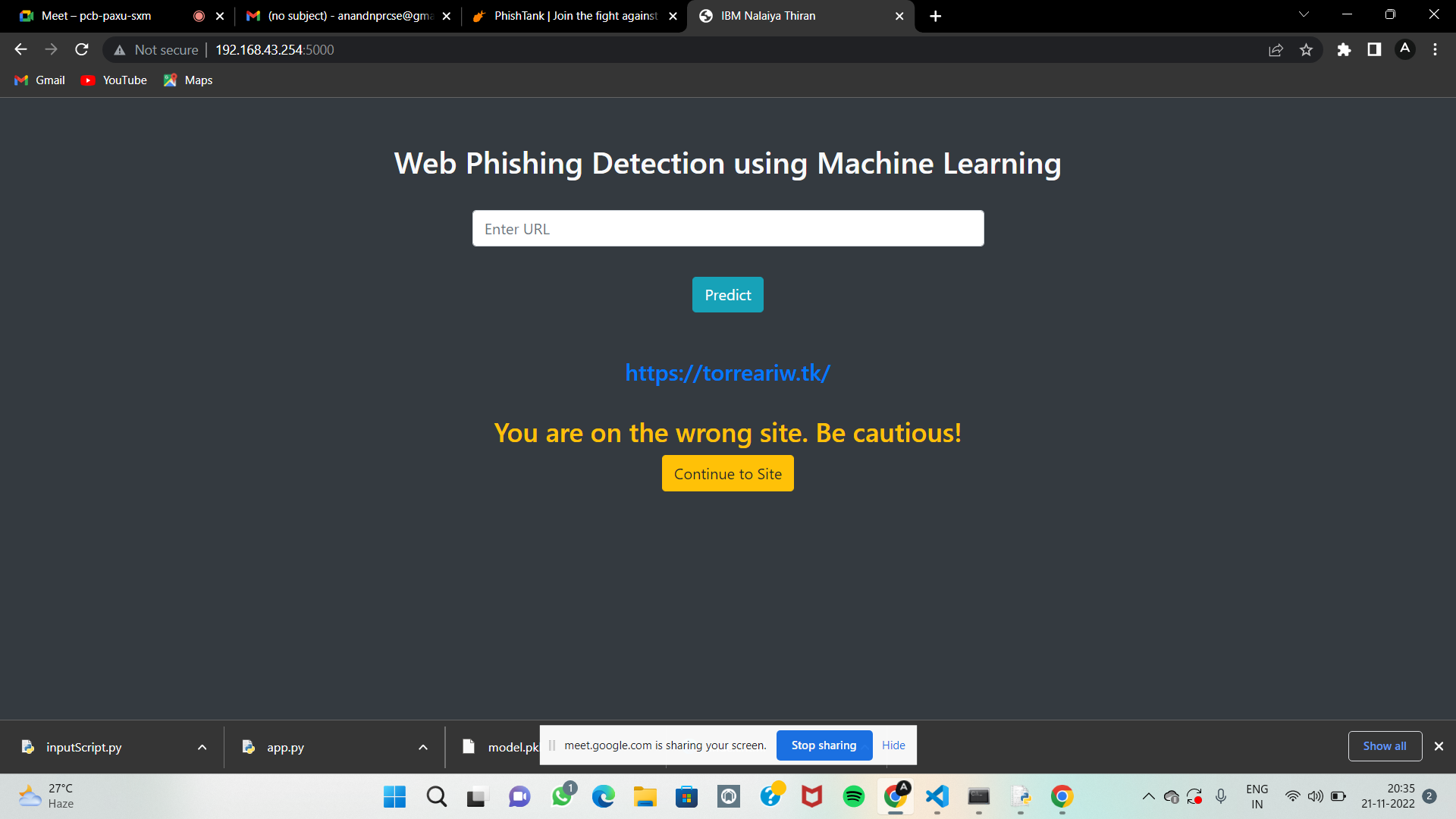
</html>

* 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.
  1. Feature 2 (Detection Result)

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







1. **TESTING** 
   1. Test Cases

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | Team ID | PNT2022TMID48697 |  | | | | | | | |
| Project Name | Project - Web Phishing Detection |
| Maximum Marks | 4 marks |
| **Test case ID** | **Feature Type** | **Component** | **Test Scenario** | **Pre-Requisite** | **Steps To Execute** | **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Commnets** | **TC for Automation(Y/N)** | **BUG ID** | **Executed By** |
| LoginPage\_TC\_OO1 | Functional | Home Page | Verify user is able to see the Homepage and check the button are works properly when user used. |  | 1.Enter URL in the box 2.Shows the predict button 3.Predicts the URL the site is Phishing or not | <https://www.primevideo.com/> | Display the Webpage and paste URL detecting the site in progress. | Working as expected | Pass |  | N |  | Team Lead & Members |
| LoginPage\_TC\_OO2 | UI | Home Page | Verify the UI elements in Login/Signup popup |  | 1.Verify homepage with below UI elements: a.URL paste box b.Predict button c.Continue site button | <https://torreariw.tk/> | All the elements are working properly and site predicts the result whether phishing or not | Working as expected | Pass |  | N |  | Team Lead & Members |

* 1. 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.

1. **Test Case Analysis**

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

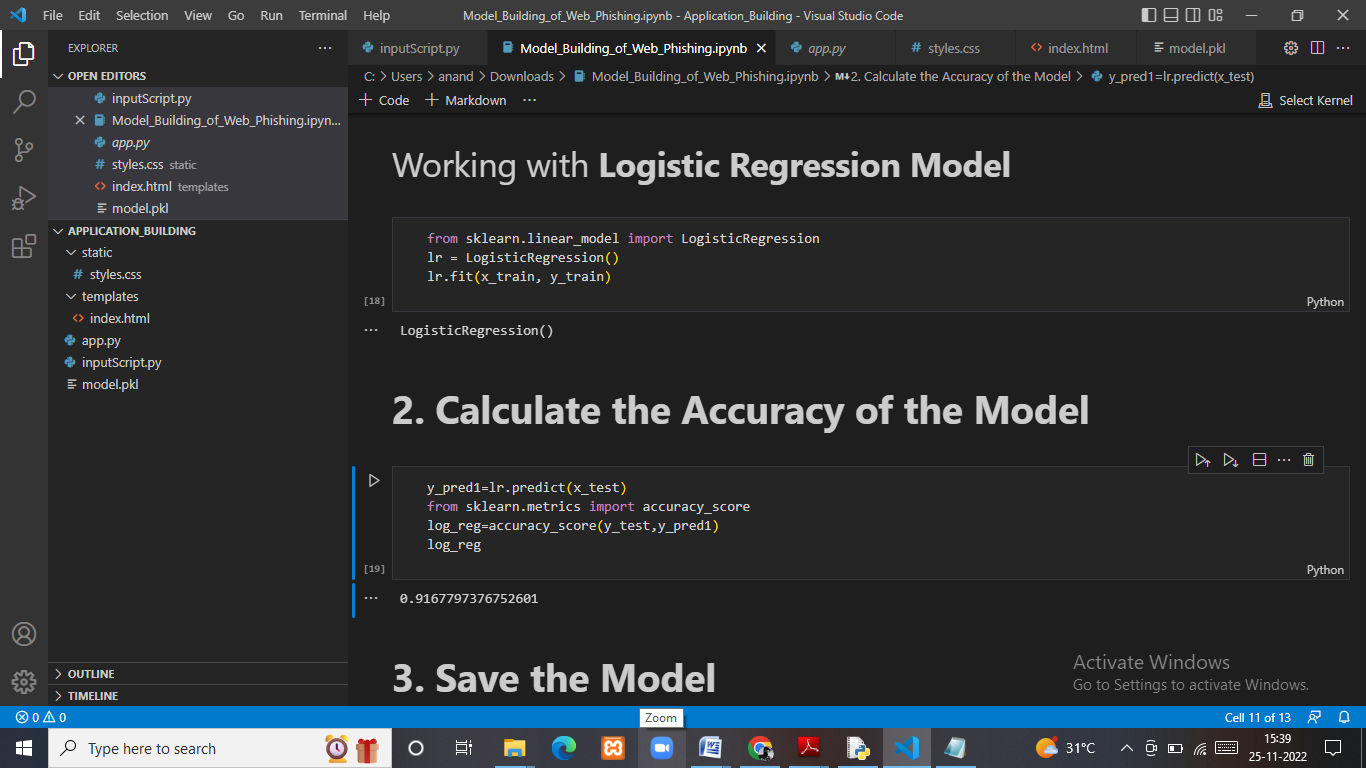
**UAT is performed by** –

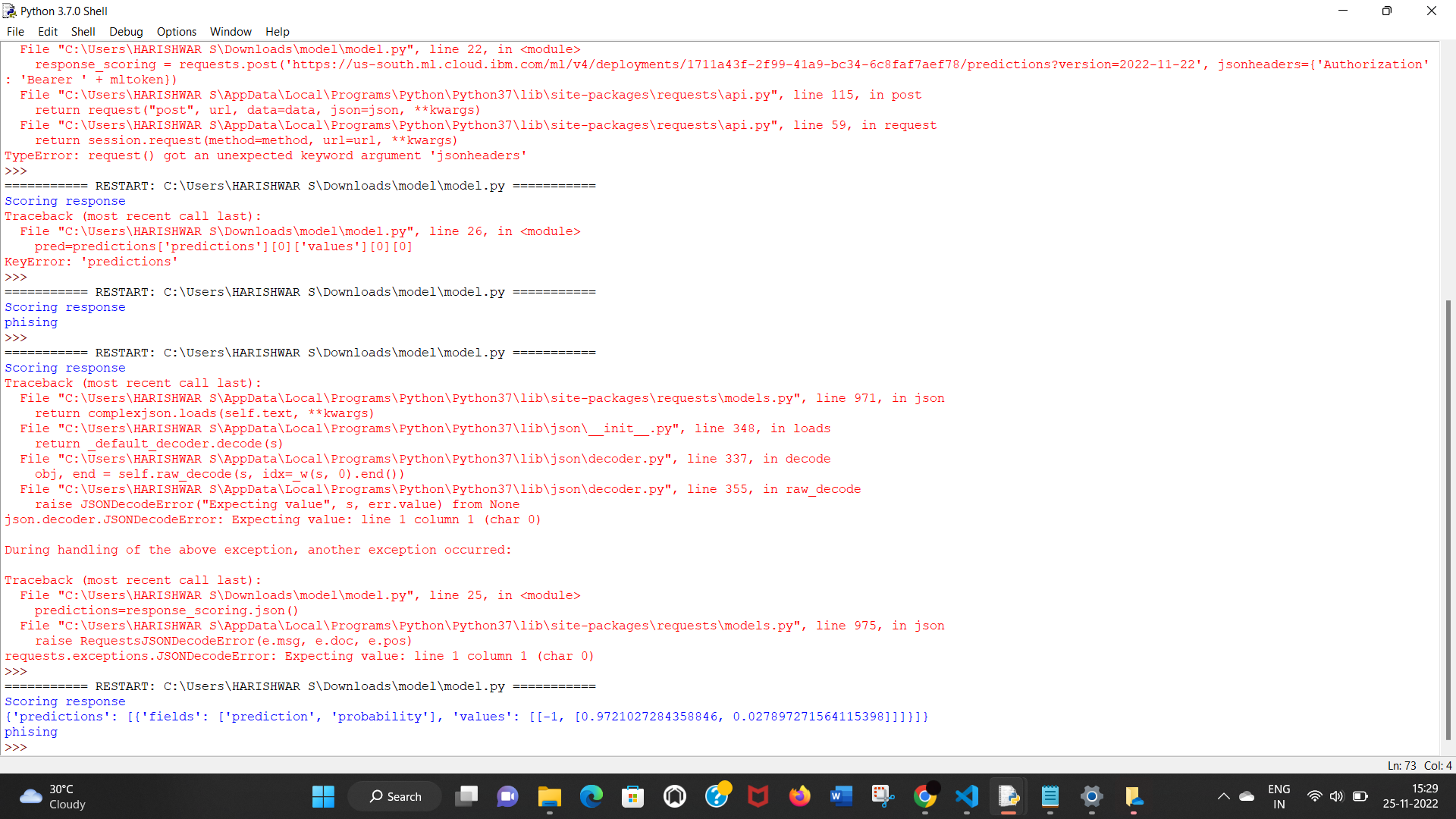
∙ Client

∙ 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.

1. **RESULTS**
   1. Performance Metrics





1. **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.

1. **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.

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 a

spoofed 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.

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 a

spoofed 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.

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 a

spoofed 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.

1. **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 ofDeep Learning, such as recurrent neural networks and convolutional neural networks can betested for increasing the performance of the system

1. **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+NcPb1dKGj7Sk" 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+OrCXaRkfj"

        crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"

        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"

        crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"

        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"

        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\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.255\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip\_address)

    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)]]

    return check

**Github**

* <https://github.com/IBM-EPBL/IBM-Project-42431-1660662531>

**Project Demo Link**

* <https://drive.google.com/file/d/149AHqIwRb9flSH8hNjFHoDUpJiWVzNtU/view?usp=share_link>

**THANK YOU**