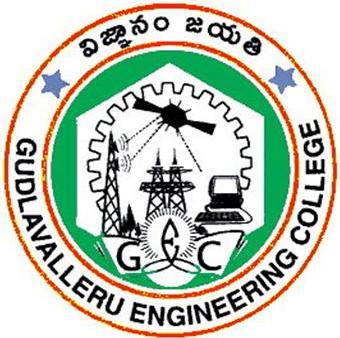


Smartinterz Guided Project

WEB PHISHING DETECTION USING IBM   
WATSON



**Team CSE\_AIML\_C02**

18481A05C6 - K. Hebsiba

18481A05E0 - Fareena Simmin

18481A05E8 - N. Sai Spandana

WEB PHISHING DETECTION USING IBM WATSON

INDEX:

1. Introduction

a. Overview

b. Purpose

2. Literature Survey

a. Existing problem

b. Proposed solution

3. Theoretical Analysis

a. Block diagram

b. Hardware / Software designing

4. Experimental Investigations

5. Flowchart

6. Result

7. Advantages & Disadvantages

8. Applications

9. Conclusion

10. Future Scope

11. Bibliography

12. Appendix

a. Source code

b. UI output Screenshot.

1. **Introduction:**

**a. Overview**

Internet has become an important part of our life to obtain, spread information in social media. While Mobile Social Networks enrich people lives, it also creates some security issues. However, 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. Phishing is the fraudulent attempt to obtain sensitive information such as username, password, bank account details and credit card details for malicious use. And phishing frauds might be the most popular cybercrime used todays. There are various domains where phishing attack can occur like online payment sector, web mail and financial institution, file hosting or cloud storage and many others. The web mail and online payment sector was targeted by phishing more than in any other industrial sector.

Phishing is the fraudulent attempt to obtain sensitive information of individuals or organization such as usernames, passwords and credit card details by disguising as trustworthy entity in an electronic communication. Phishing attack causes serious threats to user’s privacy and security. The purpose of this study is to presents an overview about web phishing attacks with urls and chose a suitable model for detecting the phishing website.

**b. Purpose**

Phishing is one of the most common and most dangerous attacks among cybercrimes. The aim of these attacks is to steal the information used by individuals and organizations to conduct transactions. Phishing websites contain various hints among their contents and web browser-based information.

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. This Project applies a machine-learning algorithm to detect Phishing websites.

1. **Literature Survey**

**a. Existing problem**

The literature search revealed several existing phishing taxonomies and anatomies. (Wetzel 2005) provides an anatomy of phishing attacks, but it ignores attack vectors and the environment where attacks occur. (Ollmann 2007a) categorizes attack initialization techniques, victim data collection techniques, and the communication media utilized in attack initialization; however, the study makes no attempt at anti-phishing techniques.

(Al Momani et al. 2013) provide a countermeasure classification schema in e-mail but ignore other attack environments. (Jakobsson and Myers 2006) provide a comprehensive view of technological countermeasures for phishing without taking into account emergent communication media and evolving attacking techniques over the past decade, while (Jakobsson and Myers 2006) were the first to comprehensively study the problem of phishing and provide a framework for studying the attack and its defences, the current research analyses countermeasures with respect to phishing techniques instead of the attack phases. For instance, (Chandrasekaran et al. 2008) classify anti-phishing approaches in website communication media into three categories, including browser plug-ins and anti-phishing toolbars, digital signing and trust propagation schemes, and content-based detection techniques. The classification not only ignores countermeasures in other types of media (e.g., e-mail, online + Social Networks), but also overlooks the dependency of counter.

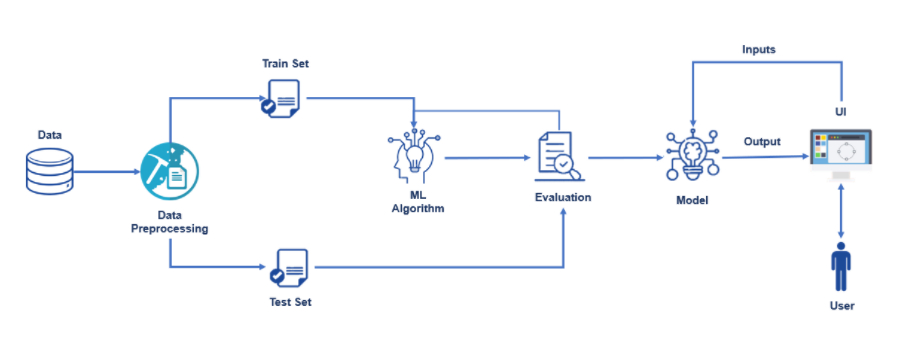
After that, the Phishing websites are detected by data mining techniques. At firstdata is collected from Phish Tank and legitimate URLs from Google. To obtain the text properties C# programming and R programming were used. 133 features were obtained from the dataset and third-party service providers. CFS subset based and Consistency subset-based feature selection methods used for feature selection and analysed with WEKA tool. Naïve Bayes and Sequential Minimal Optimization (SMO) algorithms were compared for performance evaluation and SMO is preferred by the author for phishing detection than NB. The drawback of this system is it cannot be early predicting the phishing in modern sites and it take more time for processing.

**b. Proposed solution**

Phishing are one of the most common and most dangerous attacks among cybercrimes. The aim of these attacks is to steal the information used by individuals and organizations to conduct transactions. We proposed an intelligent, flexible and effective system that is based on using classification algorithms. There are many algorithms that are used to detect the phishing websites accurately. Few of them used to classify the URL as legitimate or phished. The publicly available phishing websites data set from the machine learning repository can be used for training and testing. The features of the dataset are used to predict the result. 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.

**3. Theoretical Analysis**

**a. Block diagram**



**b. Hardware / Software designing**

**Hardware requirements:**

The following minimum basic hardware requirement was used for the implementation of the system:

* + 4 GB RAM
  + 10GB HDD
  + Intel 1.66 GHz Processor Pentium 4

**Software requirements:**

-The following software was used for the implementation of the system:

Anaconda Navigator:

* Jupyter notebook
* Spyder
* Visual Studio Code

-To build Machine learning models you must require the following packages

* **Sklearn**.
* **NumPy**
* **Pandas**
* **Matplotlib**
* **Flask**

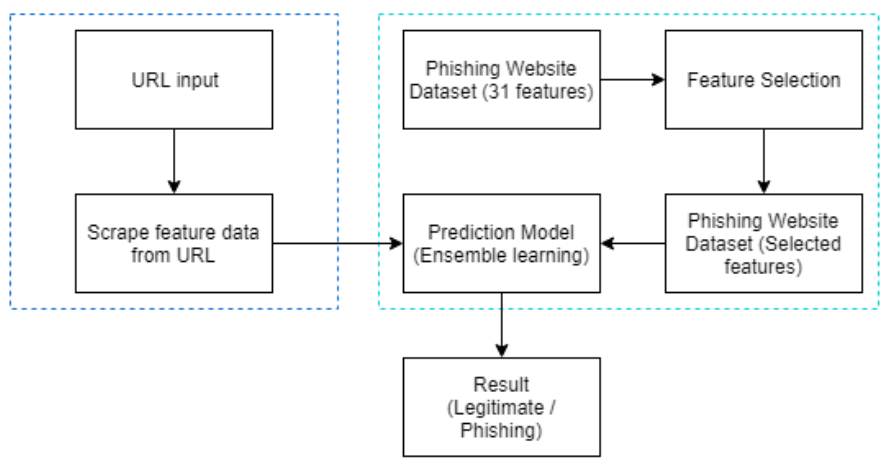
-If you are using PyCharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

**4. Experimental Investigations**

So far, we have understood that phishing is a specialized social engineering attack whereby the attacker very intelligently uses spoofed emails or websites to trick the victims into sharing their confidential and sensitive information. There is a need to understand the psychology of online consumers that whether they are concerned about the security issues when they are having the authority to change the security features. There are many academic literatures about security against phishing. However, there are a number of issues that concern the gap between academic literature and practical evidence.

A major research gap exists between research and the industry in terms of true positives. While academic and literary research essentially focuses on machine-learning and heuristics, assuming very good true positives, these true positives are sometimes high false positives. Hence, these heuristics are only reasonable enough to identify phishing sites that have not been encountered before. However, the industry primarily relies on blacklists for classification of phishing websites. But, the blacklists fail to generalize to the future unseen cases and are also potentially slow in responding to zero-hour attacks.

**5. Flowchart**



Data set: The data of URLs is obtained from Phish tank website, where Phish tank is an anti-phishing site. It contains 11055 URLs which is in structured form. Our main objective is to detect whether the URL is phishing or legitimate based on the features extracted. Where the 31 fields named *as 'having\_IPhaving\_IP\_Address', 'URLURL\_Length', 'Shortining\_Service', 'having\_At\_Symbol', 'double\_slash\_redirecting', 'Prefix\_Suffix', 'having\_Sub\_Domain', 'SSLfinal\_State', 'Domain\_registeration\_length', 'Favicon', 'port','HTTPS\_token', 'Request\_URL', 'URL\_of\_Anchor', 'Links\_in\_tags', 'SFH', 'Submitting\_to\_email', 'Abnormal\_URL', 'Redirect', 'on\_mouseover', 'RightClick', 'popUpWidnow', 'Iframe', 'age\_of\_domain', 'DNSRecord', 'web\_traffic', 'Page\_Rank', 'Google\_Index',' Links\_pointing\_to\_page', 'Statistical\_report'*

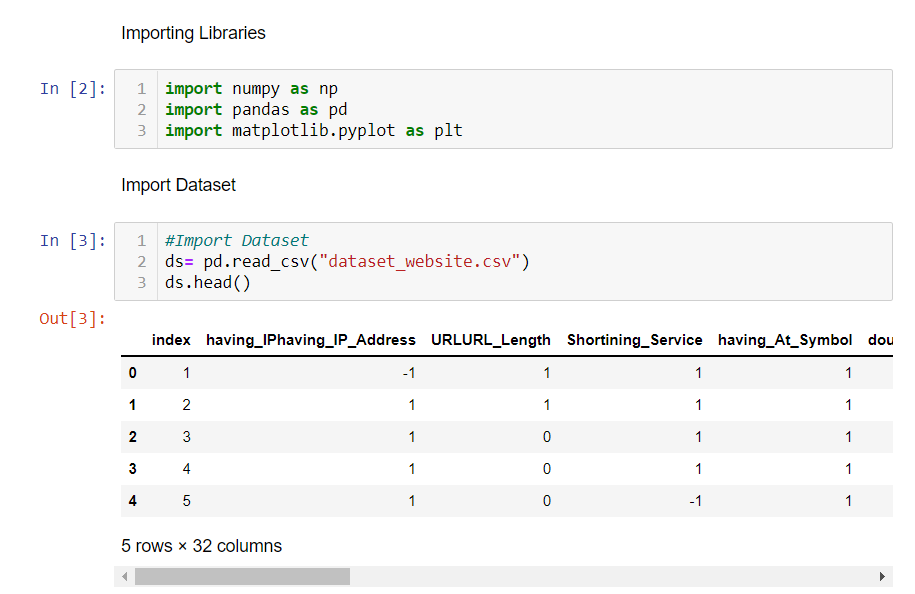
And URL input is unstructured format. In Pre-processing we have done feature extraction where The URLs are transmitted to the feature extractor, which extracts feature values through the predefined URL-based features. The features have assigned binary values 0 and 1 which indicates that feature is present or not as shown in figure below. The extracted feature values are stored as input and passed to the classifiers.

A structured dataset is given to the classifiers. We use eight methods classifications namely: Decision Tree, KNN-Manhattan Distance, KNN-Euclidean Distance, SVM, Random Forest, Logistic Regression, Naive Bayes and stacking classifier for detection of URL as phishing or legitimate. Now the classifier will find whether a requested site is a phishing site. When there is a page request, the URL of the requested site is radiated to the feature extractor. It extracts the feature values through the predefined URL-based features. These feature values are act as a input for the classifier. After this we will come to know if the site is phishing or not.

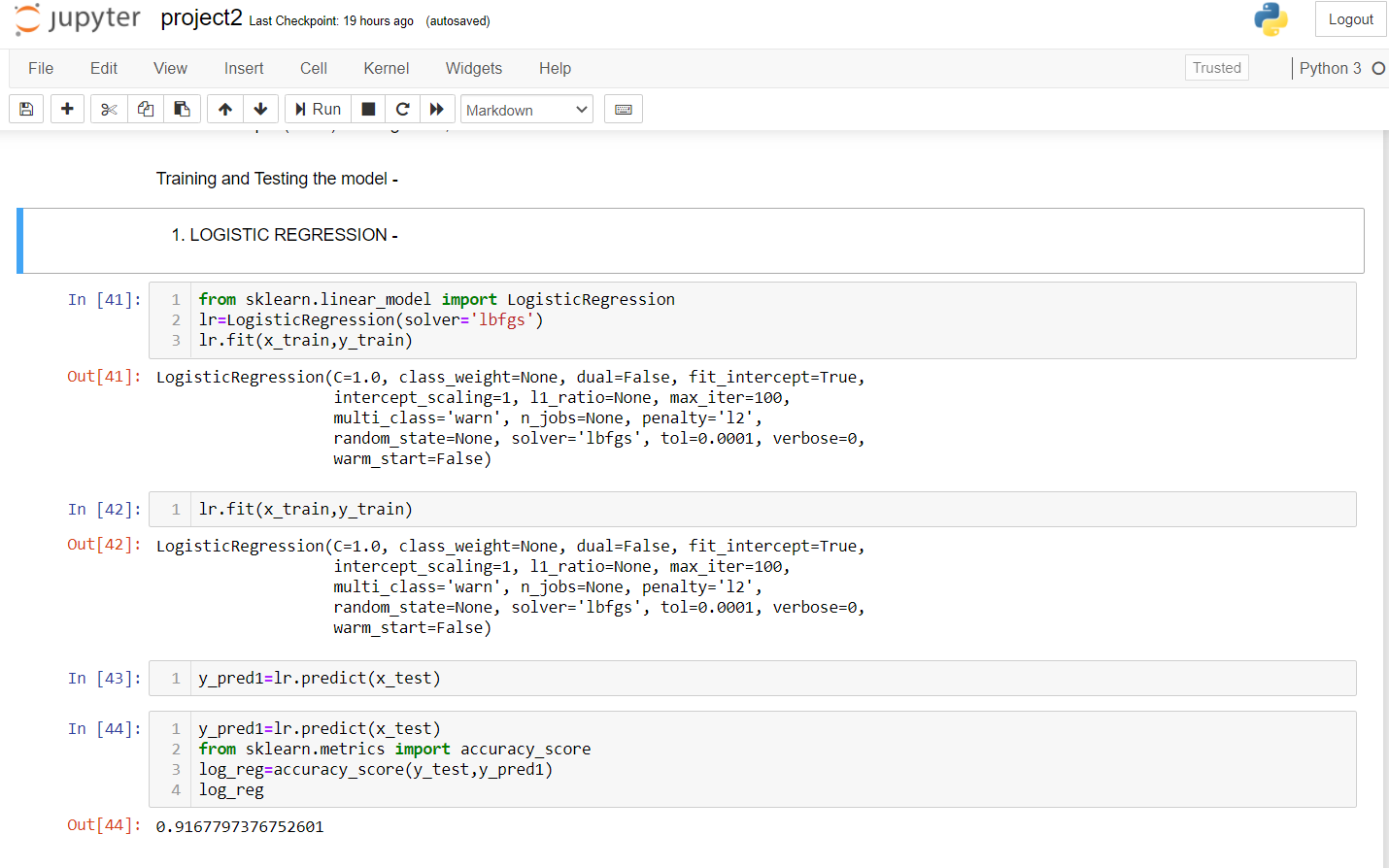
**6. Result**

- Pre-process or clean the data in dataset and analyse the pre-processed data. It includes-

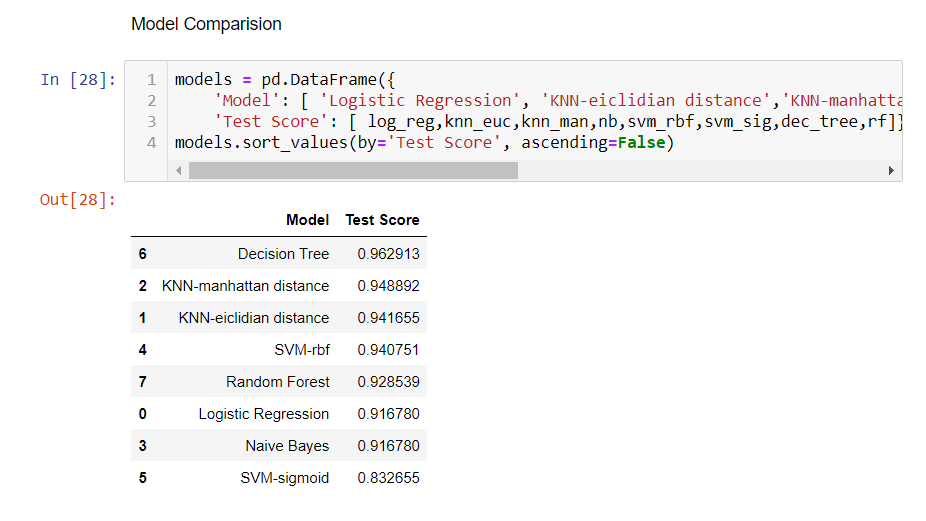
1. Handling the null values.
2. Handling the categorical values if any.
3. Normalize the data if required.
4. Identifying the dependent and independent variables.
5. Split the dataset into train and test sets

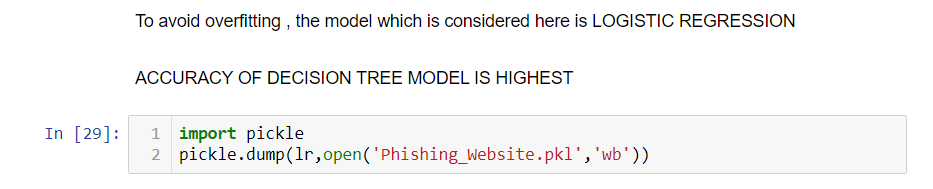


- Train the machine with pre-processed data using an appropriate machine learning algorithm.



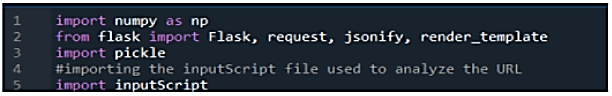
- Save the model and its dependencies.



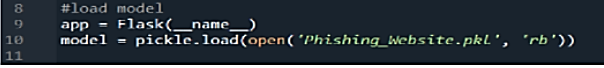


**-** Build a Web application using a flask that integrates with the model built.

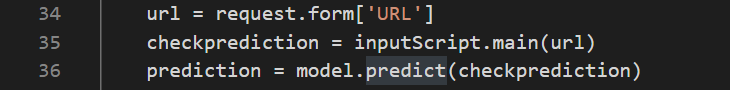
* + commands to import required libraries-



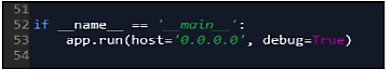
* + Load the model and initialize Flask App



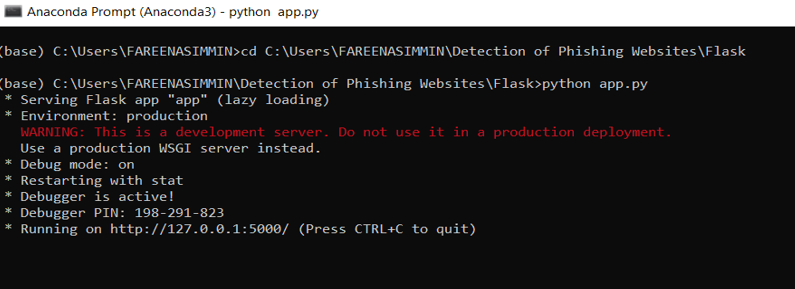
* + Fetch the URL from the UI, process the URL , get the input parameters from the URL and return the Prediction



* + Run the app



* Build An HTML Page to take the URL as a text and upon clicking on the button for submission it has to re direct to URL for “y\_predict” which returns if the URL given is Phishing or safe. The output is to be then displayed on then displayed on the page
* Execute the python code in anaconda prompt



* Now while the app is running open index.html, the above is the home page of the web application

**7. Advantages & Disadvantages**

## **Advantages:**

* The accuracy of proposed is expected to improve by effectively integrating information from multiple sources for model training.
* Reduce the processing time.
* Easily can detect whether any user given url is phishing or legitimate.

## **Disadvantages:**

* It will lead to information disclosure and property damage.
* Large organizations may get trapped in different kinds of scams.
* It cannot be early predicting the phishing in modern sites.
* It takes more time for processing

**8. Applications:**

* Phish Shield takes URL as input and outputs the status of URL as phishing or legitimate website.
* The heuristics used to detect phishing are footer links with null value, zero links in body of html, copyright content, title content and web site identify.

**9. Conclusion**

It is found that phishing attacks is very crucial and it is important for us to get a mechanism to detect it. As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using any of the machine learning algorithm with the classifier. We have seen that existing system gives less accuracy so we proposed a new phishing method that employs URL based features and also, we generated classifiers through machine learning.

“Phishing will never be eliminated, but it is important to understand this crime before proposing any solution and detect them with an efficient and most accurate score algorithm”.

**10. Future Scope**

Phishing has always had the aim of baiting users to take an action or share a piece of sensitive information by appearing as a non-threat – but awareness has since grown. Unprompted password reset emails, while once effective, no longer drive the same volume of user action and are often detected by spam filters.

Today, phishing attacks are targeted, can be difficult to detect, and grant malicious individual’s broad permissions over user data, user devices, and online services. The days of basic phishing schemes have more or less passed. Attacks now rely on advanced forms of infiltration that better disguise malicious intent.

In future if we get structured dataset of phishing, we can perform phishing detection much faster than any other technique. In future we can use a combination of any other two or more classifier to get maximum accuracy. For more accurate results, we can use Artificial Neural Network or Random Forest Classifiers. This detection tool will help to protect users from phishing attacks in the non-secured environment too. We also plan to explore various phishing techniques that uses Lexical features, Network based features, Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features from URLs and pass it through the various classifiers.

**11. Bibliography**

1. Wong, R. K. K. (2019). An Empirical Study on Performance Server Analysis and URL Phishing Prevention to Improve System Management Through Machine Learning. In Economics of Grids, Clouds, Systems, and Services: 15th International Conference, GECON 2018, Pisa, Italy, September 18-20, 2018, Proceedings (Vol. 11113, p. 199). Springer.
2. Rao, R. S., & Pais, A. R. (2019). Jail-Phish: An improved search engine based phishing detection system. Computers & Security, 83, 246-267.
3. Ding, Y., Luktarhan, N., Li, K., & Slamu, W. (2019). A keyword-based combination approach for detecting phishing webpages. computers & security, 84, 256-275.
4. Marchal, S., Saari, K., Singh, N., & Asokan, N. (2016, June). Know your phish: Novel techniques for detecting phishing sites and their targets. In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS) (pp. 323-333). IEEE.
5. Shekokar, N. M., Shah, C., Mahajan, M., & Rachh, S. (2015). An ideal approach for detection and prevention of phishing attacks. Procedia Computer Science, 49, 82-91.
6. Rathod, J., & Nandy, D. Anti-Phishing Technique to Detect URL Obfuscation.
7. Hodi, A., Kevri, J., & Karadag, A. (2016). Comparison of machine learning techniques in phishing website classification. In International Conference on Economic and Social Studies (ICESoS'16) (pp. 249-256).
8. Pujara, P., & Chaudhari, M. B. (2018). Phishing Website Detection using Machine Learning: A Review.
9. Desai, A., Jatakia, J., Naik, R., & Raul, N. (2017, May). Malicious web content detection using machine leaning. In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 1432-1436). IEEE.
10. Lakshmi, V. S., & Vijaya, M. S. (2012). Efficient prediction of phishing websites using supervised learning algorithms. Procedia Engineering, 30, 798-805.
11. Jain, A. K., & Gupta, B. B. (2018). PHISH-SAFE: URL features-based phishing detection system using machine learning. In Cyber Security (pp. 467-474). Springer, Singapore.
12. Kazemian, H. B., & Ahmed, S. (2015). Comparisons of machine learning techniques for detecting malicious webpages. Expert Systems with Applications, 42(3), 1166-1177.
13. Mao, J., Bian, J., Tian, W., Zhu, S., Wei, T., Li, A., & Liang, Z. (2019). Phishing page detection via learning classifiers from page layout feature. EURASIP Journal on Wireless Communications and Networking, 2019(1), 43.
14. Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012, December). An assessment of features related to phishing websites using an automated technique. In 2012 International Conference for Internet Technology and Secured Transactions (pp. 492-497). IEEE.
15. https://www.researchgate.net/publication/226420039-Detection-of-

Phishing-Attacks-A-Machine-Learning-Approach

**12. Appendix**

**a. Source code**

**python file – app.py :**

import numpy as np

from flask import Flask, request, jsonify, render\_template

import pickle

#Importing the inputScript file used to analyze the URL

import inputScript

import sklearn.linear\_model.logistic

import requests

# NOTE: you must manually set API\_KEY below using information retrieved from your IBM Cloud account.

API\_KEY = "QTPpKDCCko3LF2DFViJVimJGAP1dJMwrOwZgbvv2QsoM"

token\_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey": API\_KEY, "grant\_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token\_response.json()["access\_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

#load model

app = Flask(\_\_name\_\_)

model = pickle.load(open('Phishing\_Website.pkl', 'rb'))

#Redirects to the page to give the user input URL.

@app.route('/predict')

def predict():

return render\_template('final.html')

#Fetches the URL given by the URL and passes to inputScript

@app.route('/y\_predict',methods=['POST'])

def y\_predict():

# For rendering results on HTML GUI

url = request.form['URL']

checkprediction = inputScript.main(url)

payload\_scoring = {"input\_data": [{"field": ['having\_IPhaving\_IP\_Address','URLURL\_Length','Shortining\_Service','having\_At\_Symbol','double\_slash\_redirecting','Prefix\_Suffix','having\_Sub\_Domain','SSLfinal\_State','Domain\_registeration\_length','Favicon','port','HTTPS\_token','Request\_URL','URL\_of\_Anchor','Links\_in\_tags','SFH','Submitting\_to\_email','Abnormal\_URL','Redirect','on\_mouseover','RightClick','popUpWidnow','Iframe','age\_of\_domain','DNSRecord','web\_traffic','Page\_Rank','Google\_Index',' Links\_pointing\_to\_page','Statistical\_report'],

"values": checkprediction }]}

response\_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/feb0a218-dcf0-4857-af8d-5738137d9f78/predictions?version=2021-07-04', json=payload\_scoring, headers={'Authorization': 'Bearer ' + mltoken})

pred = response\_scoring.json()

pred = model.predict(checkprediction)

print("Scoring response")

print(pred)

output=pred

if(output==1):

pred="Your are safe!! This is a Legitimate Website."

else:

pred="You are on the wrong site. Be cautious!"

return render\_template('final.html', prediction\_text='{}'.format(pred),url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions

@app.route('/predict\_api',methods=['POST'])

def predict\_api():

# For direct API calls trought request

data = request.get\_json(force=True)

prediction = model.y\_predict([np.array(list(data.values()))])

output = prediction[0]

return jsonify(output)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', debug=True)

**final.html (/predict):**

<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

<meta charset="UTF-8">

<title>Prediction</title>

<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/final1.css') }}">

<link rel="stylesheet" href="C:\Users\FAREENASIMMIN\Detection of Phishing Websites\Flask\static\css\final1.css">

<style>

.login{

top: 20%;

}

</style>

</head>

<body>

<div class="header">

<div>NILA</div>

<ul>

<li><a href="C:\Users\FAREENASIMMIN\Detection of Phishing Websites\Flask\index.html#contact">Contact</a></li>

<li><a href="C:\Users\FAREENASIMMIN\Detection of Phishing Websites\Flask\index.html#about">About</a></li>

<li><a href="C:\Users\FAREENASIMMIN\Detection of Phishing Websites\Flask\index.html">Home</a></li>

</ul>

</div>

<div class="main">

<h1>Phishing Website Detection using Machine Learning<h1>

</div>

<form action="{{ url\_for('y\_predict')}}"method="post">

<input type="text" name="URL" placeholder="Enter the URL to be verified" required="required" />

<button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>

</form>

<br>

<br>

<div id='result',class='result' style='color:black;font-size:30px;'>{{ prediction\_text }}</div>

<a href=" {{ url }} "> {{ url }} </a>

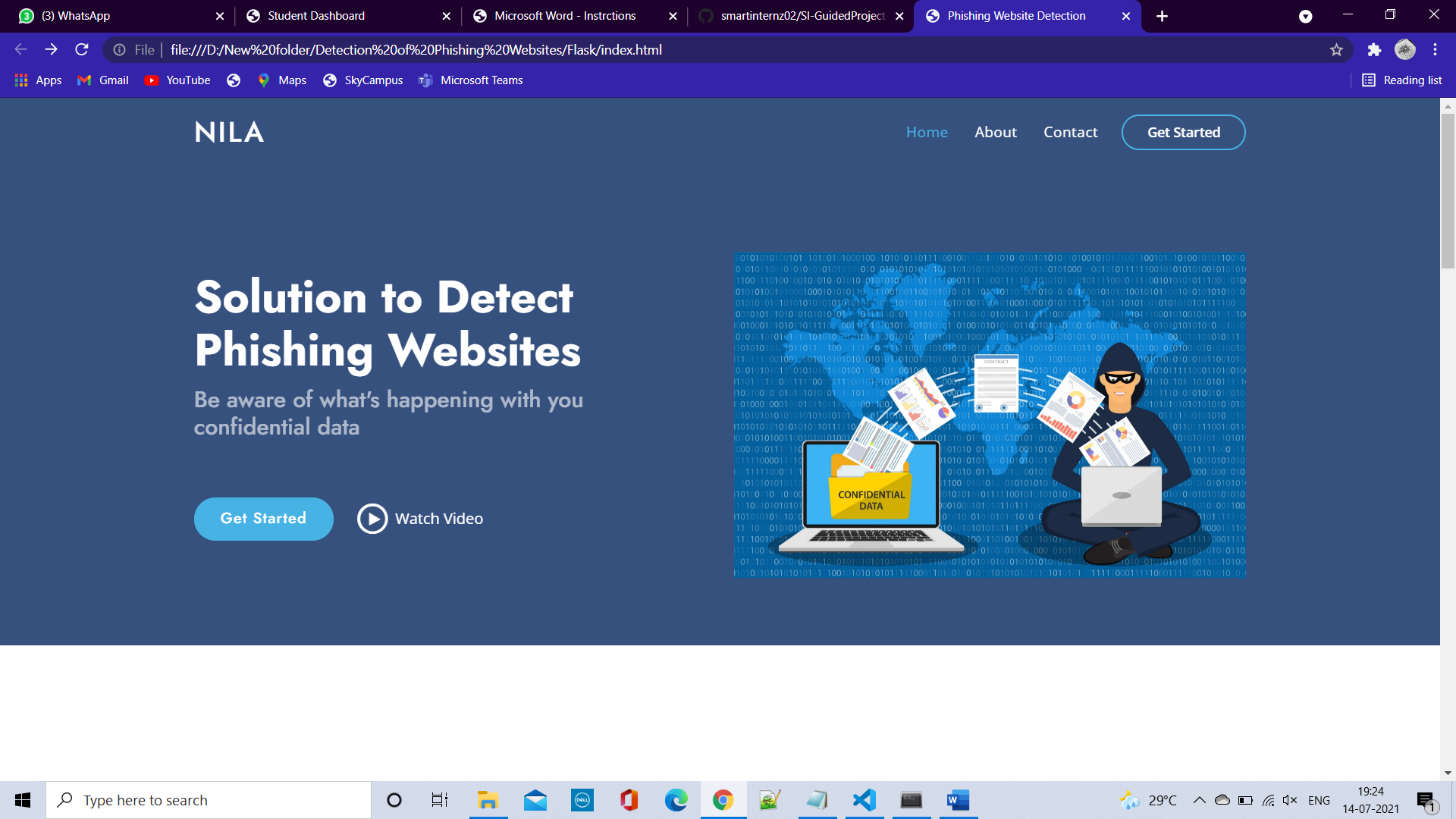
<!-- Template Main JS File -->

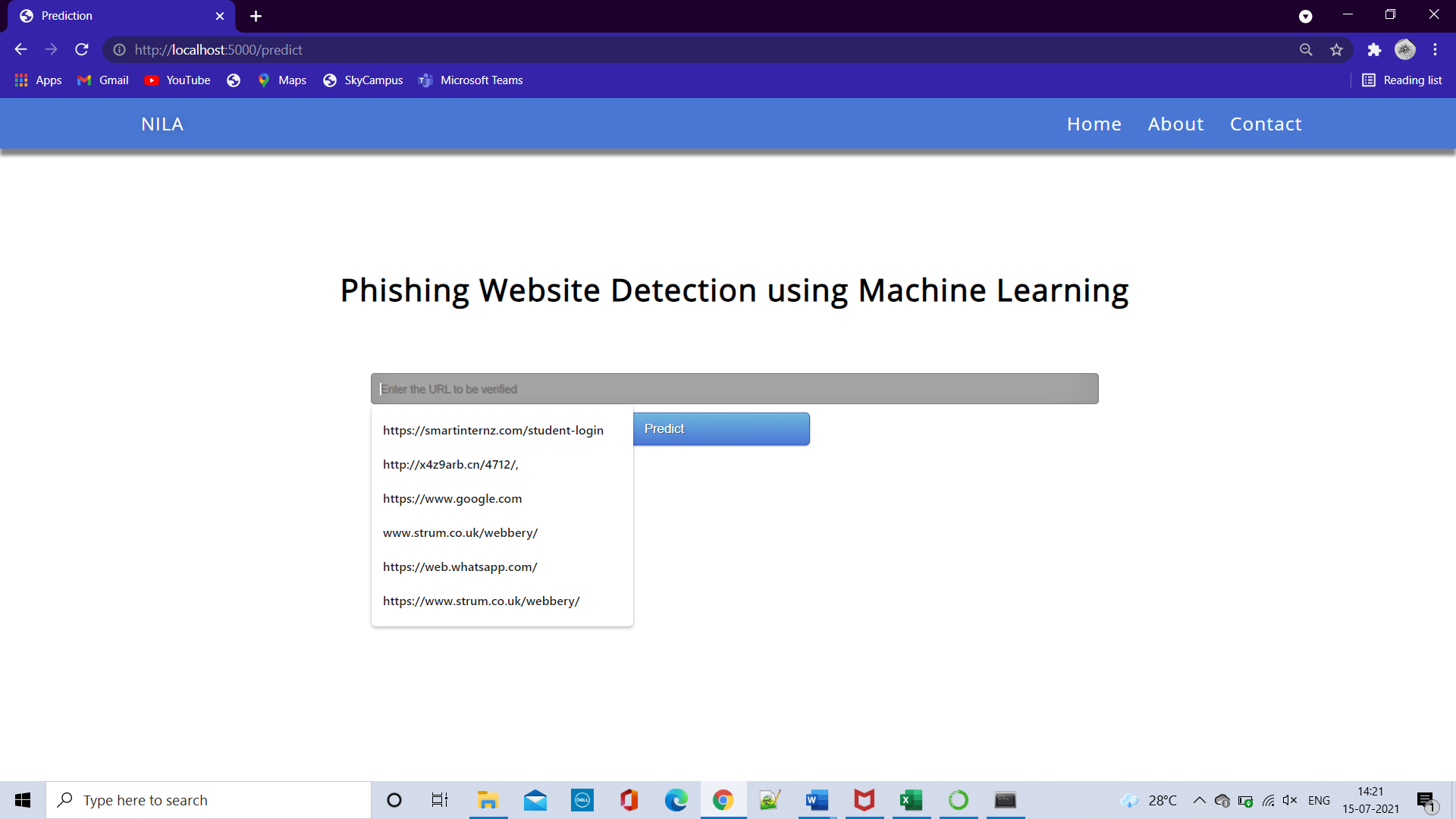
<script src="assets/js/main.js"></script>

</body>

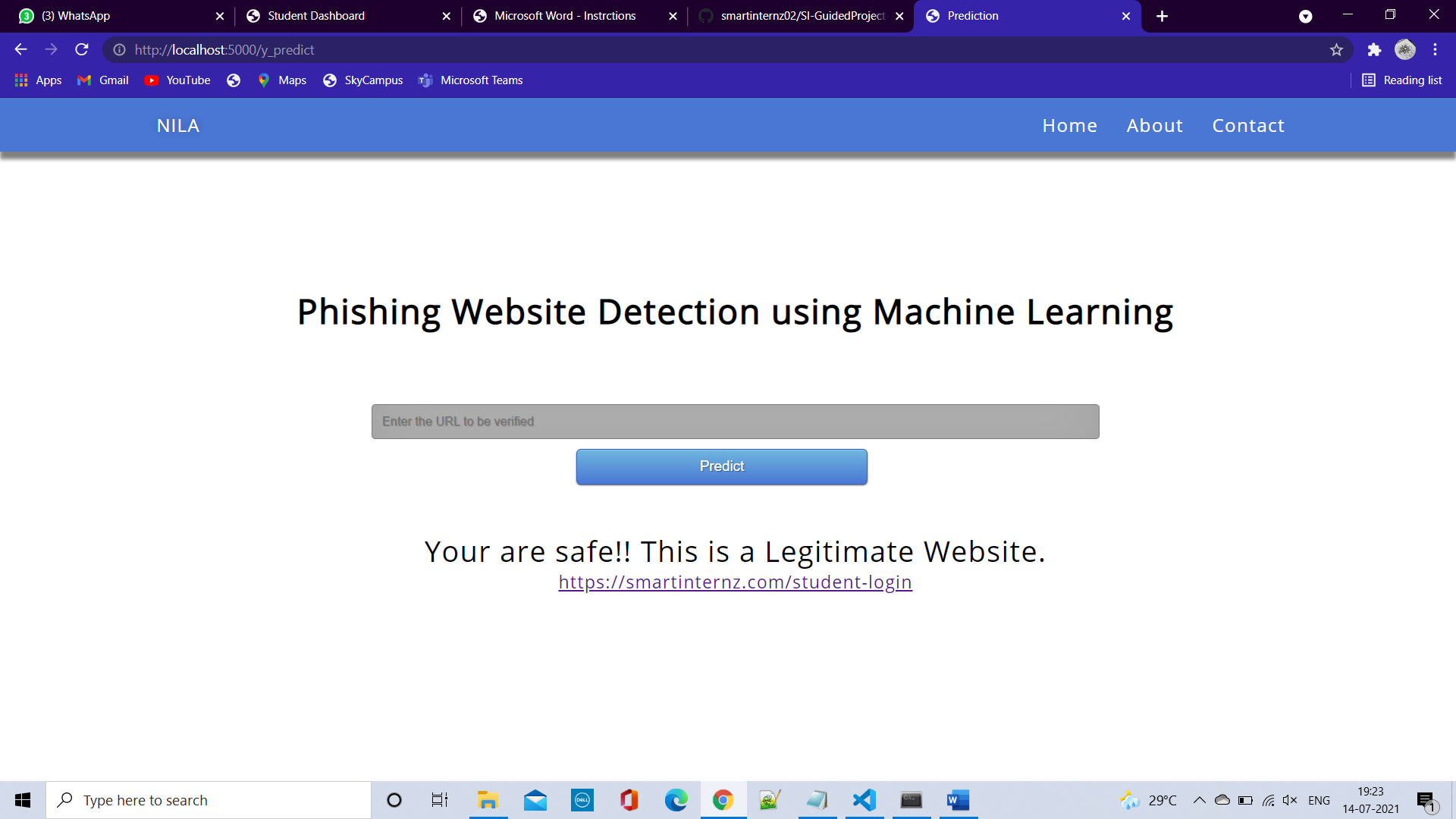
</html>

**b. UI output Screenshot**





* Example for Legitimate website URL:





* Example for Phishing website URL:

