**SONA COLLEGE OF TECHNOLOGY**

PHISHING OF WEBSITES URLs USING MACHINE LEARNING

TECHNOLOGGY IN IBM WATSON STUDIO.

PROJECT DONE BY:

Bhuvanesh.D

Arun Karthick.P

Hiruthik.K

**INTRODUCTION:**

* 1. Overview

Phishing is one of the most severe cyber-attacks where researchers are interested to find a solution. In phishing, attackers lure end-users and steal their personal in-formation. To minimize the damage caused by phishing must be detected as early as possible. There are various phishing attacks like spear phishing, whaling, vishing, smishing, pharming and so on. There are various phishing detection techniques based on white-list, black-list, content-based, URL-based, visual-similarity and machine-learning. In this paper, we discuss various kinds of phishing attacks, attack vectors and detection techniques for detecting the phishing sites. Performance comparison of 18 different models along with nine different sources of datasets are given. Challenges in phishing detection techniques are also given.

1.2) Purpose

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](https://www.sciencedirect.com/topics/computer-science/phishing-attack) 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](https://www.sciencedirect.com/topics/computer-science/website-phishing-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

2) **LITERATURE SURVEY**

2.1) Existing Service

Phishing is popular among attackers, since it is easier to trick someone into clicking a malicious link which seems legitimate than trying to break through a computer’s defense systems. The malicious links within the body of the message are designed to make it appear that they go to the spoofed organization using that organization’s logos and other legitimate contents.

2.2) Proposed Solution

The proposed system has mainly six modules. They are

➢ Dataset Collection

➢ Image pre-processing

➢ Model Building

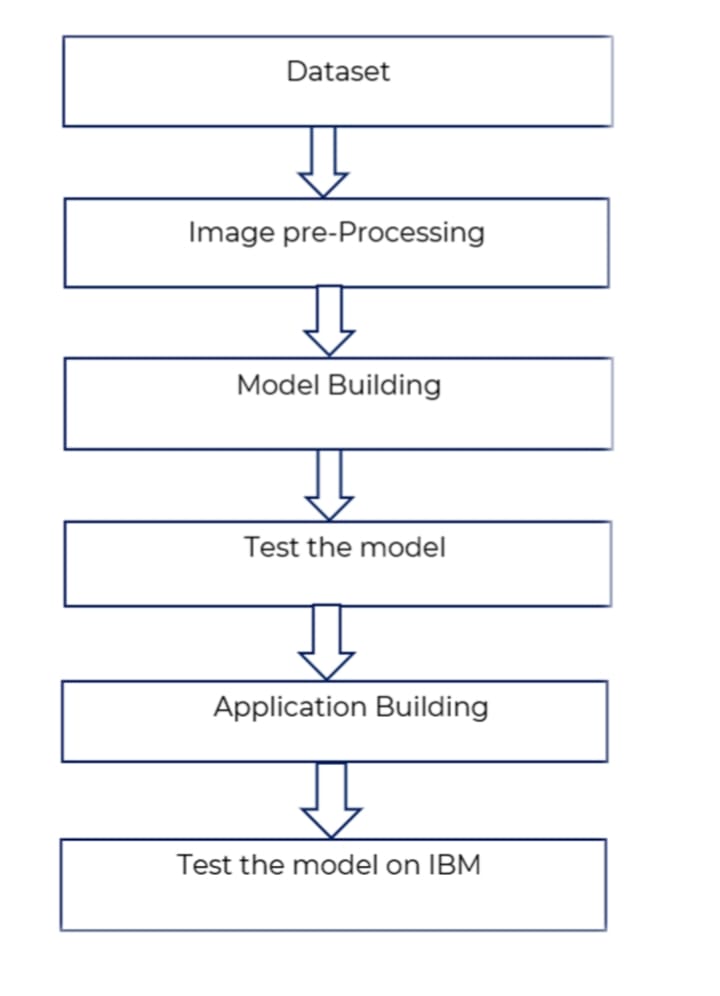
➢ Test the model

➢ Application Building

➢ Test the model on IBM

**3) THEORITICAL ANALYSIS**

**3.1) Block Diagram**

****

**3.2 .1 ) Software designing**

**Anaconda Navigator :**

Anaconda is a free and open-source distribution of the Python and R programming

languages for scientific computing that aims to simplify package management and deployment.

Package versions are managed by the package management system conda. The Anaconda

distribution includes data-science packages suitable for Windows, Linux, and macOS.

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda

package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a

graphical alternative to the command-line interface (CLI).

**Jupyter Notebook :**

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the

conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator,

as a graphical alternative to the command line interface (CLI). A Jupyter Notebook document

is a JSON document, following a versioned schema, and containing an ordered list of

input/output cells which can contain code, text mathematics, plots and rich media, usually

ending with the “. ipynb" extension.

**Tensor flow :**

TensorFlow is an end-to-end open-source platform for machine learning. It has a

comprehensive, flexible ecosystem of tools, libraries, and community resources that lets

researchers push the state-of-the-art in ML and developers can easily build and deploy ML

powered applications.

**Keras :**

Keras is an open-source neural-network library written in Python. It is capable of

running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed

to enable fast experimentation with deep neural networks, it focuses on being user-friendly,

modular, and extensible. Keras contains numerous implementations of commonly used neural-

network building blocks such as layers, objectives, activation functions, optimizers, and a host

of tools to make working with image and text data easier to simplify the coding necessary for

writing deep neural network code.

**Flask :**

Flask is a microframework written in python. It is classified as

a microframework because it does not require particular tools or libraries.It has no database

abstraction layer, form validation, or any other components where pre-existing third-party

libraries provide common functions.

3.2 .2 ) **Hardware designing**

• Processor: Intel core i5 or above.

• 64-bit, quad-core, 2.5 GHz minimum per core

• Ram: 8 GB or more

• Hard disk: 10 GB of available space or more.

• Display: Dual XGA (1024 x 768) or higher resolution monitors

• Operating system: Window

**4) EXPERMENTAL INVESTIGATIONS**

The analysis of the project that would be developed by our hands. It consists of six steps

where the execution starts from taking an input as website from the data set followed by the data

pre-processing, model building,testing the model and then Save the model and its

dependencies,Building a Web application using flask that integrates with the model built using

IBM Watson studio. Finally, the output is observed after all the above mentioned steps are

completed.

**4.1) Dataset collection :**

The data set used is has been downloaded from Kaggle which consists of many website that are used to test and train the system.

**4.3) Model Building :**

The neural network model is to be built by adding different network layers like convolution,

pooling, flattening, dropout and neural layers.

For this step we need to import Keras and other packages that we’re going to use in

building the CNN. Import the following packages: Sequential is used to initialize the neural

network. Adding Convolution layer is used to make the convolutional network that deals with

the images. Adding Pooling layer is used to add the pooling layers. Flatten layer is the function

that converts the pooled feature map to a single column that is passed to the fully connected

layer. Adding fully connected layer which includes hidden layer to the neural network.

**SEQUENTIAL:**

To initialize the neural network, we create an object of the Sequential class.

Ex; model=Sequential()

**4.4) Test the model :**

The model is to be tested with different website to know if it is predicting correctly. . In

split the data we set the website as 80% Training Data and 20% Testing Data. Then build CNN

model train deep neural network for epochs

**4.5) Application Building**

After the model is built, we will be integrating it into a web application so that normal

users can also use it. The users need to give the scan to know if the tumor is present or not.

**4.6) Test the model in IBM**

At finally test the model in IBM using IBM Watson studio and then build a ‘.h5’

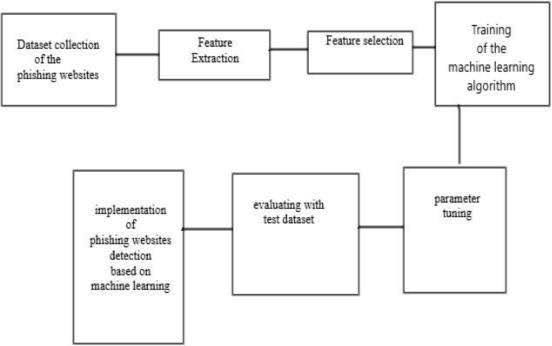
using this file we can predict weather a patient is effected brain tumor or not in web

application using flask

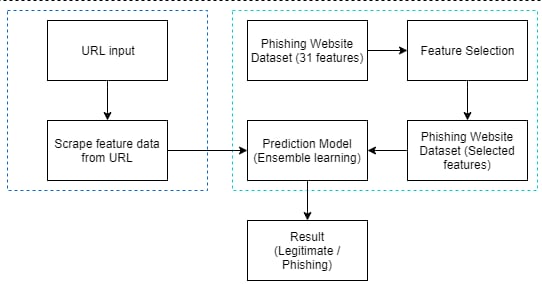
Finally , we detect whether the given MRI brain website is safe or not using IBM Watson Studio.

**5) FLOWCHART**

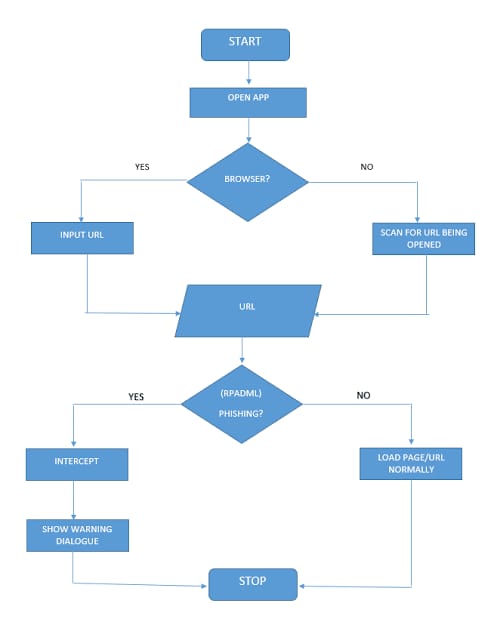
5.1) Block Diagram



5.2) Flow Diagram

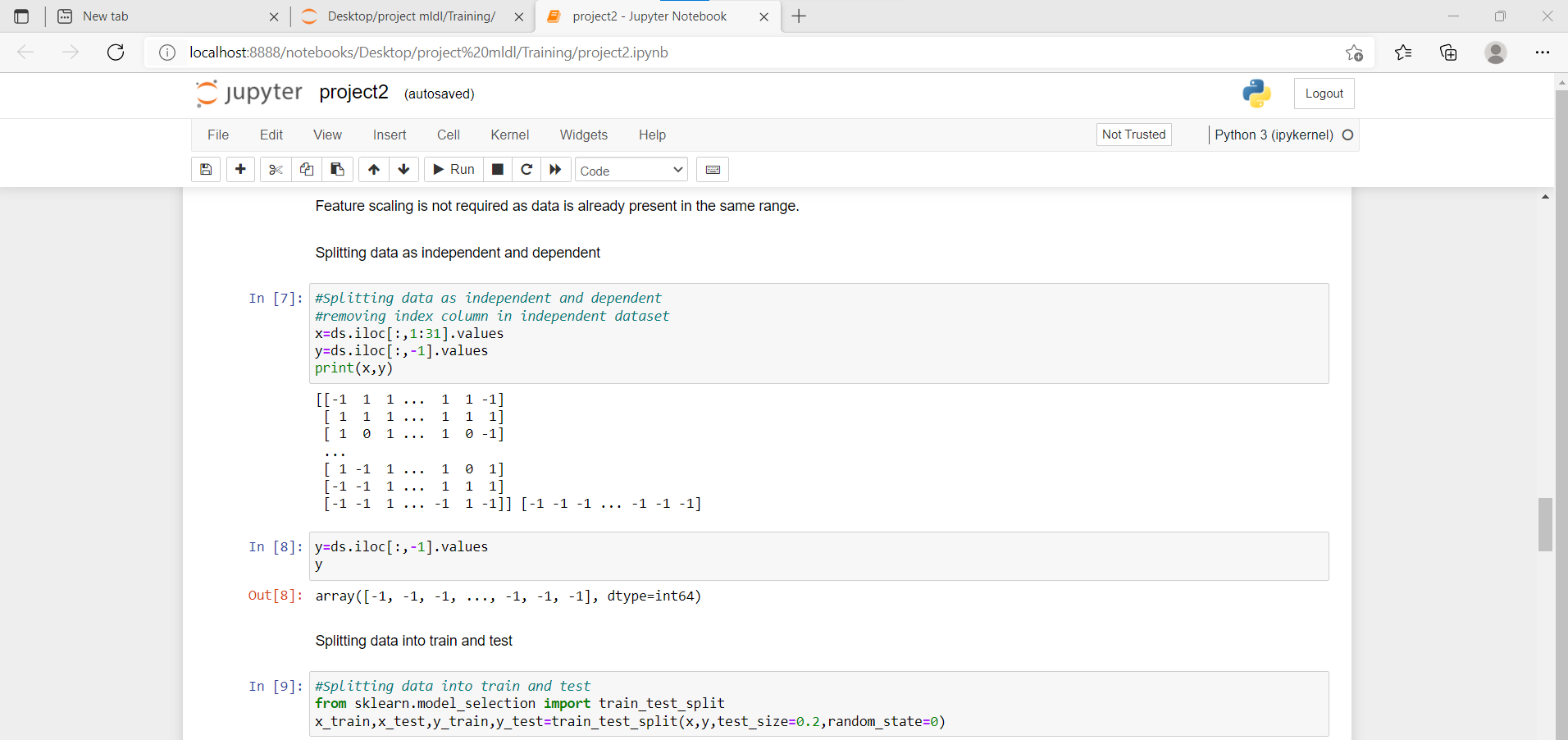


5.3) Use Case Diagram

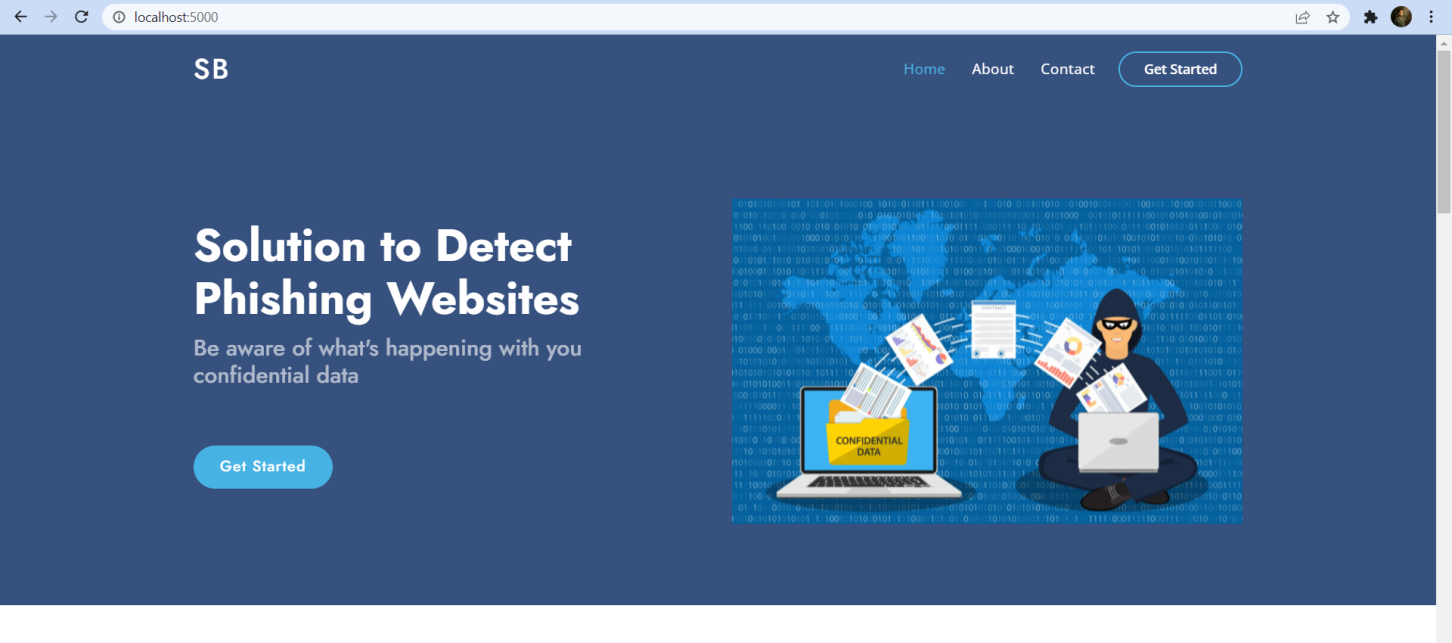


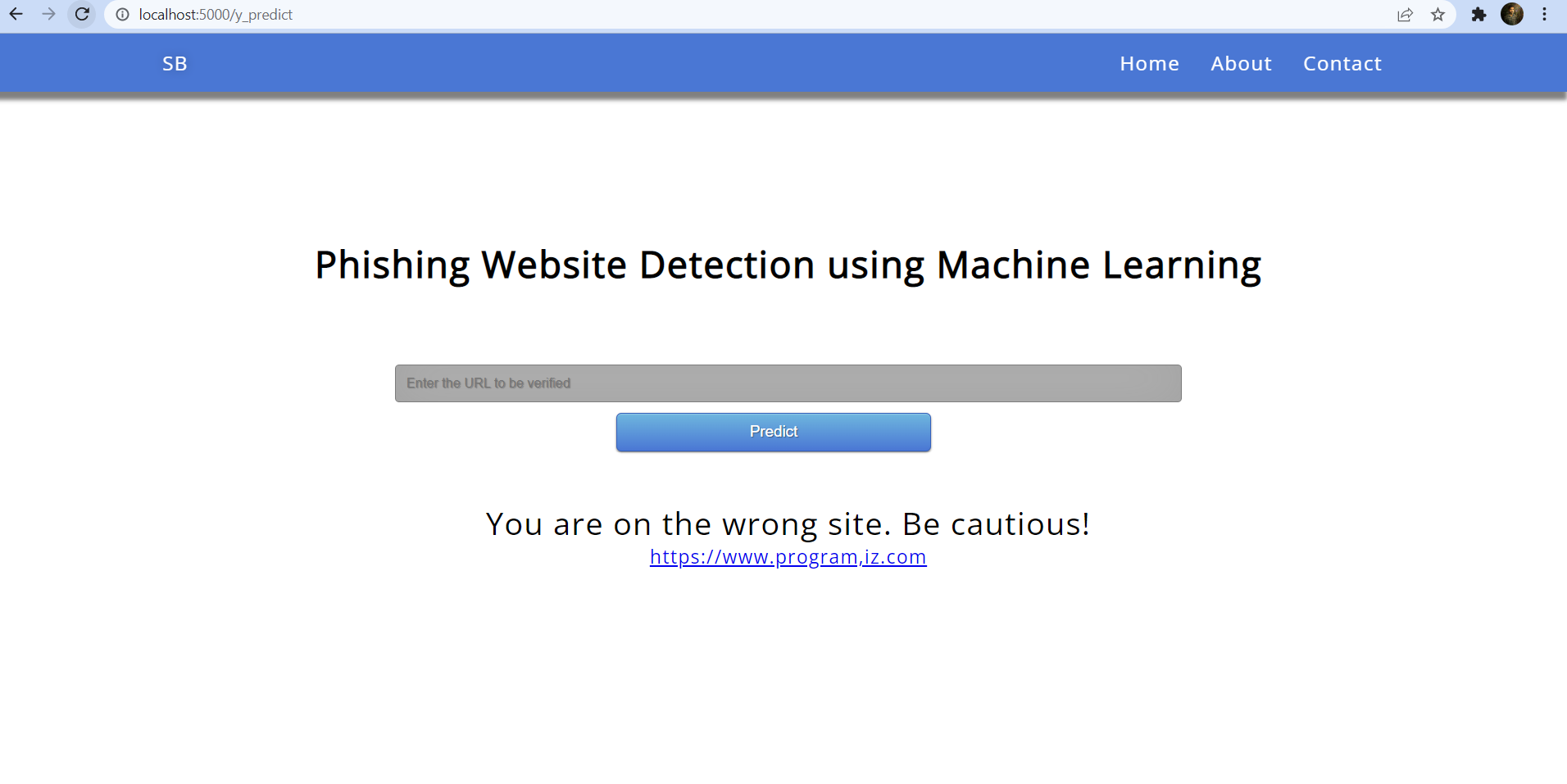
**6) RESULT:**

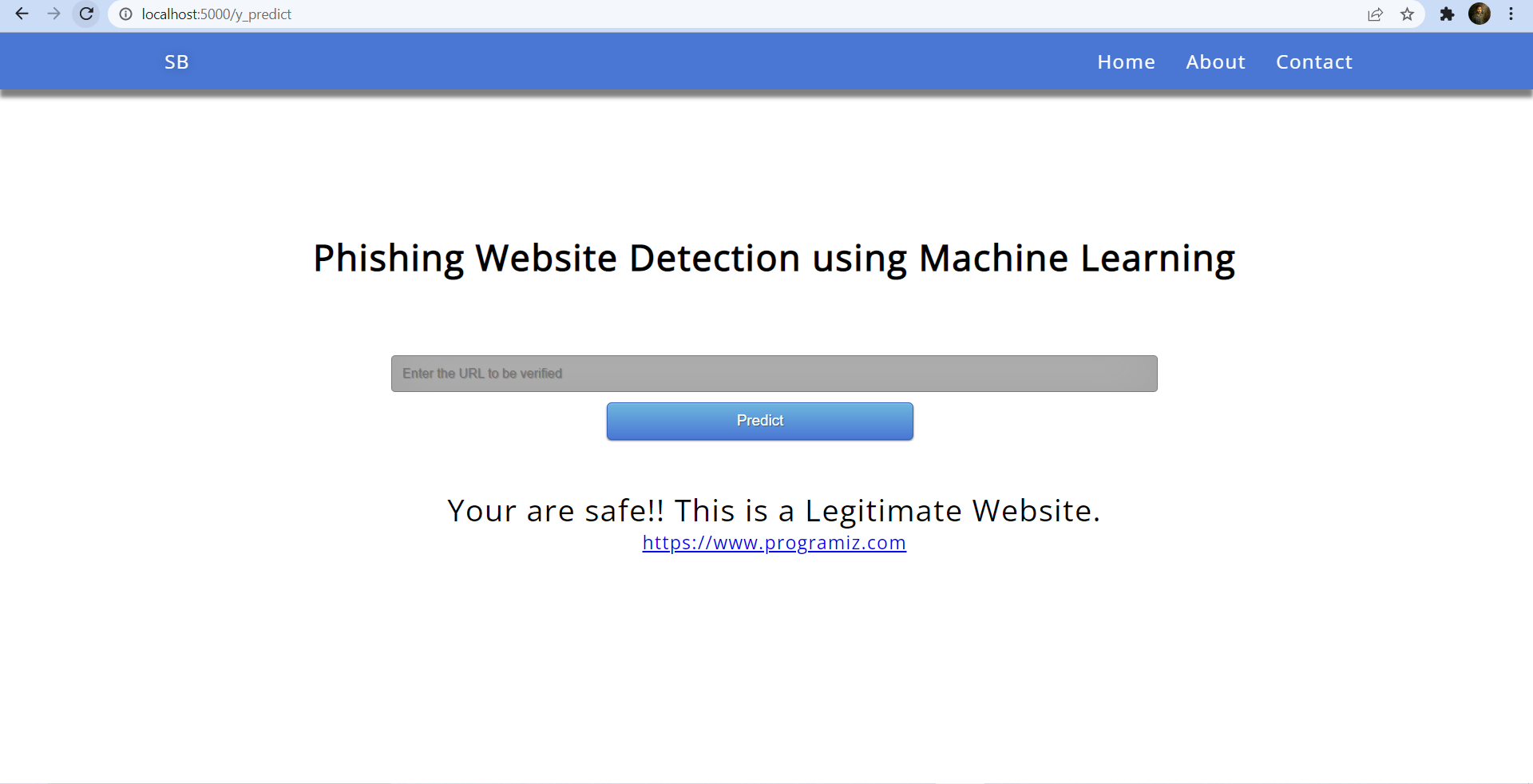
Train & Test the data



Flask Imlementation:







7) Advantage & Disadvantage:

Advantage

* Build secure connection for the user to use
* Provide clear idea about the website’s Privacy

Disadvantage

* Time consuming
* Consuming memory
* High Cost

**8) APPLICATION:**

➢ The main aim of the applications is Detection of phishing website.

➢ The main reason behind the development of this application is to provide user’s safety.

➢ This application is helpful to user’s.

➢ It is user friendly application

**9) Conclusion:**

phishing websites by several researchers in Machine

Learning. On reviewing the papers, we came to a conclusion

that most of the work done by using familiar machine learning algorithms like Naïve Bayesian, SVM, Decision Tree and Random Forest. Some authors proposed a new system like PhishScore and PhishChecker for detection. The combinations of features with regards to accuracy, precision, recall etc. were used. Experimentally successful techniques in detecting phishing website URLs were summarized in Table 1. As phishing websites increases day by day, some features may be included or replaced with new ones to detect them.

**10) Future Scope:**

This paper develops detection system with a wide protection scope using URL features only which is relying on the fact that users directly deal with URLs to surf the internet and provides a good approach to detect malicious URLs as proved by previous studies. Additionally, Anti-phishing solutions can be positioned at different levels of attack flow where most researchers are focusing on client side solutions which turn to add more processing overhead at the client side and lead to losing the trust and satisfaction of the users. Nowadays many organizations make centralized protection of spam filtering. This paper proposes a system which can be integrated into such process in order to increase the detection performance in a real time. The simulation results of the proposed system showed a phishing URLs detection accuracy with 93% and provided online process of a single URL in average time of 0.12 second.

**11) Source code:**

import numpy as np

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

import pickle

#importing the inputScript file used to analyze the URL

import inputScript

#load model

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

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

@app.route('/')

def helloworld():

return render\_template("index.html")

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

@app.route('/predict')

def predict():

return render\_template('final.html')

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

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

def y\_predict():

'''

For rendering results on HTML GUI

'''

url = request.form['URL']

checkprediction = inputScript.main(url)

print(checkprediction)

prediction = model.predict(checkprediction)

print(prediction)

output=prediction[0]

if(output==1):

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

else:

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

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

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

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

def predict\_api():

'''

For direct API calls trought request

'''

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

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

output = prediction[0]

return jsonify(output)

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

app.run(debug=False)