URL – Based Phishing Detection Using Machine Learning

ABSTRACT

Phishing URLs mainly target individuals and/or organizations through social engineering attacks by exploiting the humans’ weaknesses in information security awareness. These URLs lure online users to access fake websites, and harvest their confidential information, such as debit/credit card numbers and other sensitive information. In this work, we introduce a phishing detection technique based on URL lexical analysis and machine learning classifiers. The experiments were carried out on a dataset that originally contained 1056937 labeled URLs (phishing and legitimate). This dataset was processed to generate 22 different features that were reduced further to a smaller set using different features reduction techniques. Random Forest, Gradient Boosting, Neural Network and Support Vector Machine (SVM) classifiers were all evaluated, and results show the superiority of SVMs, which achieved the highest accuracy in detecting the analyzed URLs with a rate of 99.89%. Our approach can be incorporated within add-on/middleware features in Internet browsers for alerting online users whenever they try to access a phishing website using only its URL.

In recent years, advancements in Internet and cloud technologies have led to a significant increase in electronic trading in which consumers make online purchases and transactions. This growth leads to unauthorized access to users’ sensitive information and damages the resources of an enterprise. Phishing is one of the familiar attacks that trick users to access malicious content and gain their information. In terms of website interface and uniform resource locator (URL), most phishing webpages look identical to the actual webpages. Various strategies for detecting phishing websites, such as blacklist, heuristic, Etc., have been suggested. However, due to inefficient security technologies, there is an exponential increase in the number of victims. The anonymous and uncontrollable framework of the Internet is more vulnerable to phishing attacks. Existing research works show that the performance of the phishing detection system is limited. There is a demand for an intelligent technique to protect users from the cyber-attacks. In this study, the author proposed a URL detection technique based on machine learning approaches. A recurrent neural network method is employed to detect phishing URL. Researcher evaluated the proposed method with 7900 malicious and 5800 legitimate sites, respectively. The experiments’ outcome shows that the proposed method’s performance is better than the recent approaches in malicious URL detection.

CHAPTER – 1
INTRODUCTION

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.

Common threats of web phishing:

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

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

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.

1.1 Motivation

- Detecting and preventing phishing websites square measure continually a vital space of analysis. Different types of phishing techniques offer torrential and essential ways that for effectively police work and, protective the counselling of the people and organizations.

- Uniform resource locator plays a vital role in phishing.

- Uniform resource locator may be a major space, through that the websites are initiated and thru the link the pages square measure redirected to following page.

- Redirecting the pages is that the vulnerable construct in phishing (i.e.) through the hyperlink; the pages square measure redirected to the egitimate web site or the phishing site.

1.2 Problem 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.

CHAPTER – 2
AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 Aim

A URL based phishing attack is carried out by sending malicious links, that seems legitimate to the users, and tricking them into clicking on it. In phishing detection, an incoming URL is identified as phishing or not by analysing the different features of the URL and is classified accordingly. Different machine learning algorithms are trained on various datasets of URL features to classify a given URL as phishing or legitimate.

2.2 Scope

The phishing website will appear the same as the legitimate website and directs the user to a page to enter personal details of the user on the fake website. Through machine learning algorithms one can improve the accuracy of the prediction. The proposed method predicts the URL based phishing attacks based on features and also gives maximum accuracy. This method uses uniform resource locator (URL) features. We identified features that phishing site URLs contain.

CHAPTER – 3
EXPERIMENTAL OR MATERIALS AND METHODS : ALGORITHMS USED

3.1 Pre Requisites

In order to develop this project we need to install the following software/packages:

Step 1:
Anaconda Navigator :
Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

For this project, we will be using Jupyter notebook and Spyder

Step 2:

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

Open anaconda prompt.

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

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

3.2 Project Objectives

By the end of this project:

- You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- Applying different algorithms according to the dataset
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

3.3 Project Flow

Find below the project flow to be followed while developing the project.

- Download the dataset.
- Preprocess or clean the data.
- Analyze the pre-processed data.
- Train the machine with preprocessed 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.

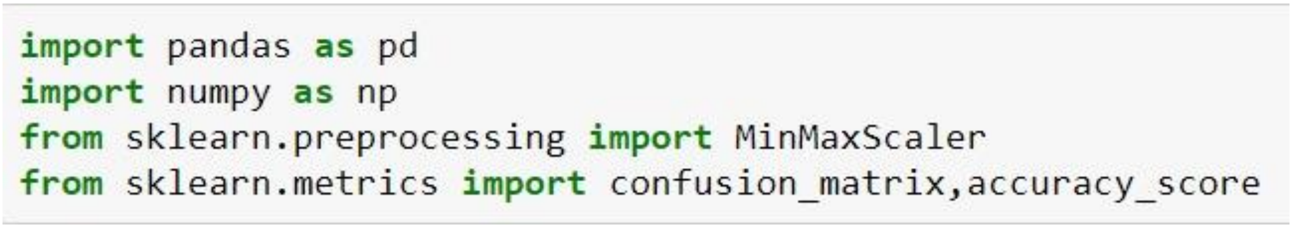
3.4 Data Pre-Processing

Import Required Libraries

Step 1 - Launch Jupyter notebook through anaconda navigator or anaconda prompt.

Step 2 - Create a new notebook by clicking on "new" button on the top right corner of the page.

The libraries can be imported using the import keyword. Insert commands as shown below.



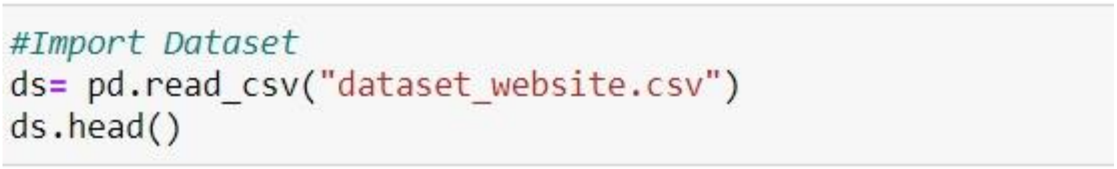
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
```

Fig-3.4.1: Importing required libraries

Read The Dataset

The dataset is read as a dataframe by using pandas library. Insert the commands as shown below

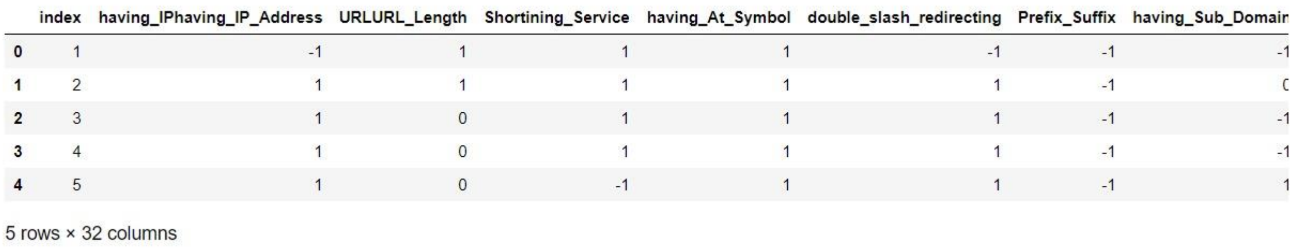
(Here ds is referred as dataframe & pd is the alias name given to pandas library).



```python
#Import Dataset
ds= pd.read_csv("dataset_website.csv")
ds.head()
```

Fig-3.4.2: Reading the Dataset

Sample output of the dataset rows is shown below.



	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domair
0	1	-1	1	1	1	-1	-1	-1
1	2	1	1	1	1	1	-1	0
2	3	1	0	1	1	1	-1	-1
3	4	1	0	1	1	1	-1	-1
4	5	1	0	-1	1	1	-1	1

5 rows × 32 columns

Fig-3.4.3: Sample Output of Dataset

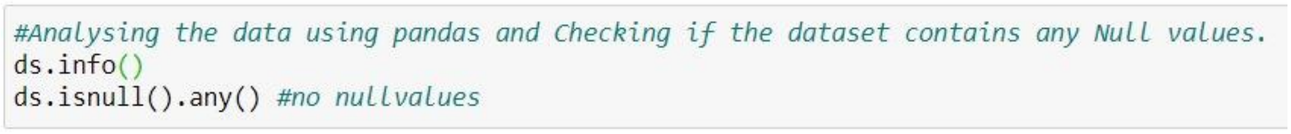
Handling Null Values

Checking for Null values in a dataset and handling if any

In this activity, we will check if there are any null values in a dataset and fill/handle them.

To know if there are any null values present in a dataset isnull() method can be used.

Input the commands as shown below to check for null values.



```
#Analysing the data using pandas and Checking if the dataset contains any Null values.
ds.info()
ds.isnull().any() #no nullvalues
```

Fig-3.4.4: Handling the Null values

Output: The command ds.isnull().any() returns true if null values are present.

Here, the dataset which we have used doesn’t have any null values.

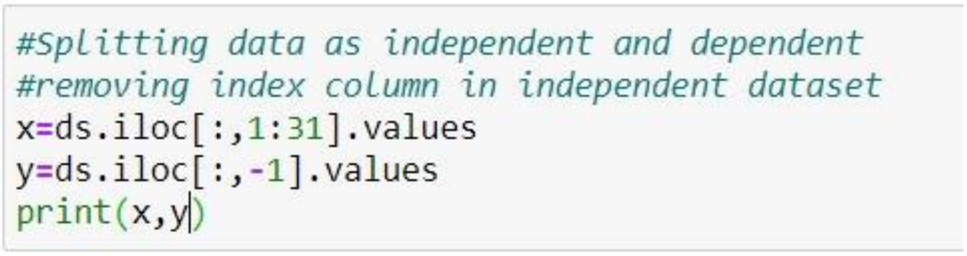
Splitting The Data

Splitting data into independent and dependent variables

Identifying Independent & dependent variables:

In this activity, the dependent and independent variables are to be identified. The last column (Result) in the dataset is the dependent variable which is dependent on the 30 different factors. The independent columns are considered as x and the dependent column as y

Input the commands as shown



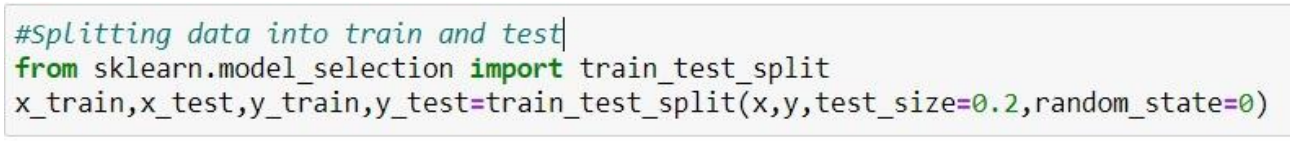
```
#Splitting data as independent and dependent
#removing index column in independent dataset
x=ds.iloc[:,1:31].values
y=ds.iloc[:,-1].values
print(x,y)
```

Fig-3.4.5: Splitting the Dataset

Splitting the data:

After identifying the dependent and independent variables, the dataset now has to be split into two sets, one set is used for training the model and the second set is used for testing how good the model is built. The split ratio we consider is 80% for training and 20% for testing.

Input the commands as shown



```
#Splitting data into train and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

Fig-3.4.6: Splitting the Test and Train Data

3.5 Imported Necessary libraries

To build Machine learning models you must require the following packages

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

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

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

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

Flask: Web framework used for building Web applications.

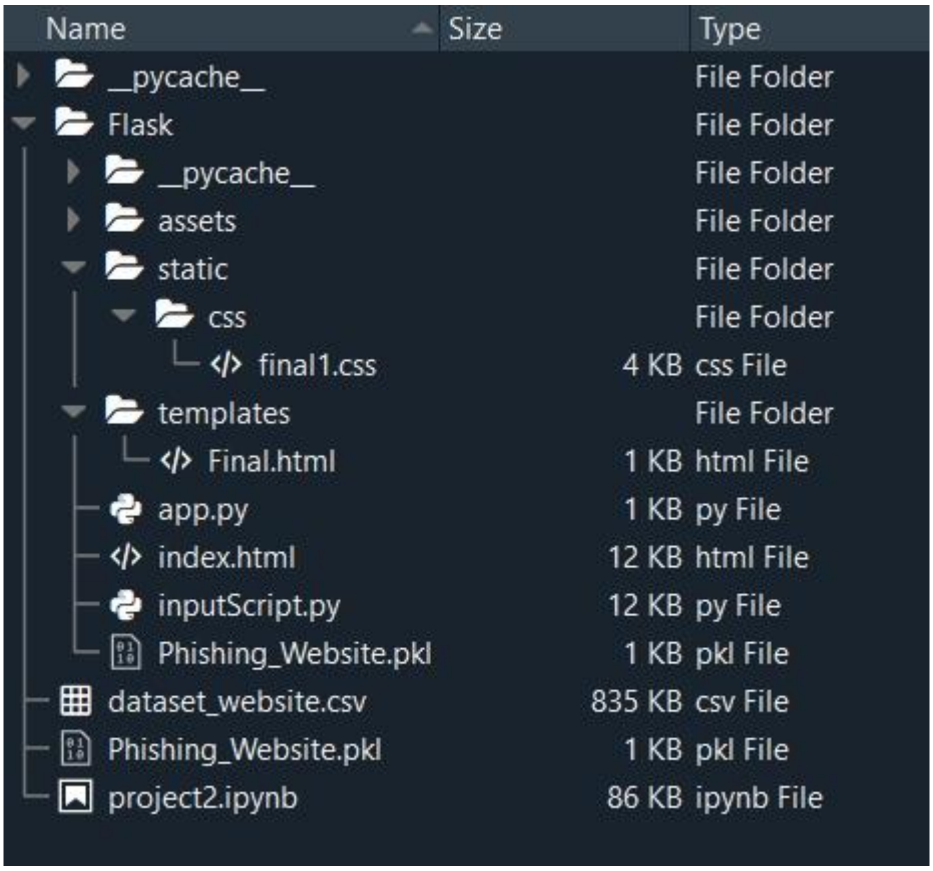
Watch the video below to learn how to install packages.

CHAPTER – 4
RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

4.1 Project Structure

- "Flask" folder contains two sub-folders static and templates. Static folder contains the style sheet used.
- "Templates" folder has the HTML pages.
- "app.py" is the python script for server side computing.
- "index.html" is the home page which should be used for initiating the application.
- "inputScript.py" has all the parameters of evaluation for a URL.
- "Phishing_Website.pkl" is the model file which you have to build.
- "dataset_website.csv" is the dataset
- "project2.ipynb" is the training notebook

Refer below image for the same:



Name	Size	Type
__pycache__		File Folder
Flask		File Folder
__pycache__		File Folder
assets		File Folder
static		File Folder
css		File Folder
final1.css	4 KB	css File
templates		File Folder
Final.html	1 KB	html File
app.py	1 KB	py File
index.html	12 KB	html File
inputScript.py	12 KB	py File
Phishing_Website.pkl	1 KB	pkl File
dataset_website.csv	835 KB	csv File
Phishing_Website.pkl	1 KB	pkl File
project2.ipynb	86 KB	ipynb File

Fig-4.1: Project Structure

4.2 Model Building

Choose The Appropriate Model

Working with Logistic Regression model

Step – 1: Here, We will be initially considering Logistic Regression model and fit the data.

Step – 2: Check the metrics of the model. Here we will be evaluating the model built. We use the test set for evaluation. The test set is given to the model for prediction and prediction values are stored in another variable called y_pred1.

The actual and predicted values are compared to know the accuracy of the model using the accuracy_score function from sklearn.metrics package.

Follow the below steps to find the accuracy of the model.

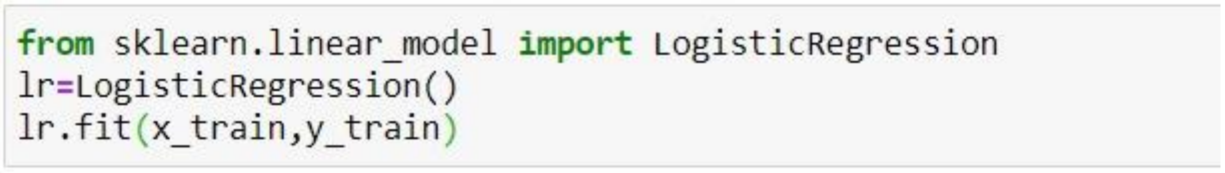
The accuracy for logistic regression model for this dataset is 91.6%.

Note: You can use different classification models to know the performance and choose whichever works better.

Step – 3: Saving the model**.**

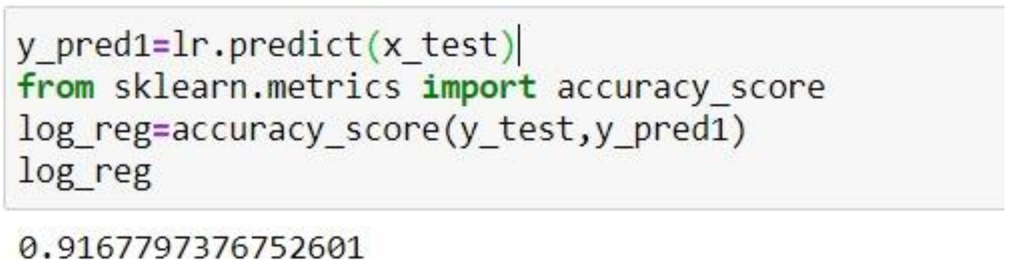
The finalized model is now to be saved. We will be saving the model as a pickle or pkl file.

Use the command below to save the model.



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

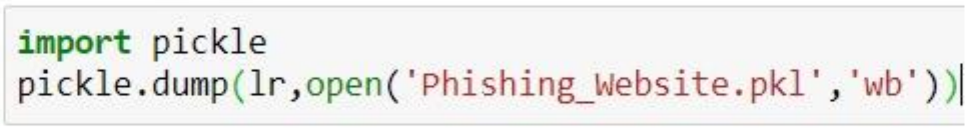
Fig-4.2.1: Working with logistic regression model



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

```
0.9167797376752601
```

Fig-4.2.2: Accuracy



```
import pickle
pickle.dump(lr,open('Phishing_Website.pkl','wb'))
```

Fig-4.2.3: Saving Model as pkl file

4.3 Application Building

Building an Application to integrate the model

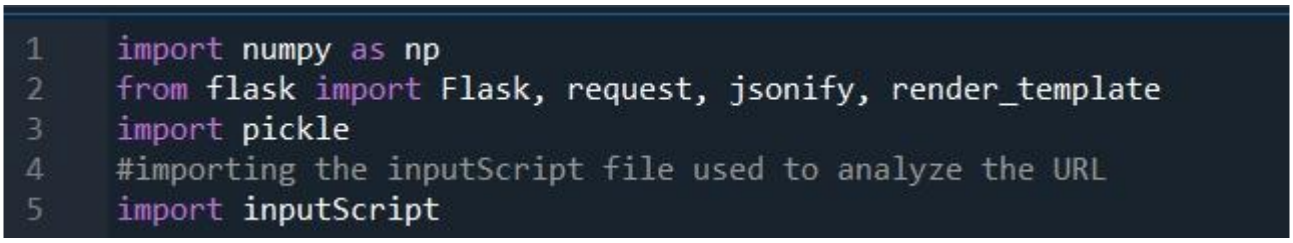
After the model is built, we will be integrating it to a web application so that normal users can also use it to know if any website is phishing or safe in a no-code manner.

In the application, the user provides any website URL to check and the corresponding parameter values are generated by analysing the URL using which legitimate websites are detected.

Flask App
Step – 1: Build the python flask app

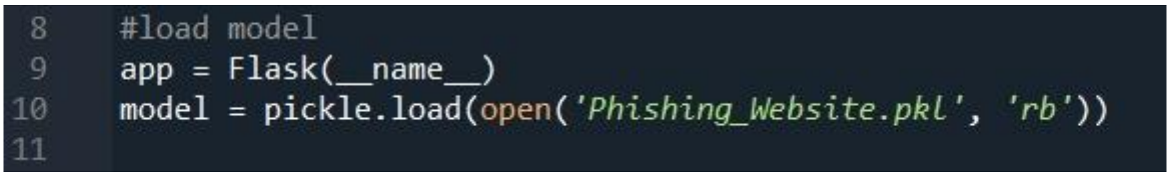
In the flask application, the URL is taken from the HTML page and it is scraped to get the different factors or the behavior of the URL. These factors are then given to the model to know if the URL is phishing or safe and is sent back to the HTML page to notify the user.

Input the following commands to Import required libraries



```python
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
```

Fig-4.3.1: Load the model and initialize Flask App

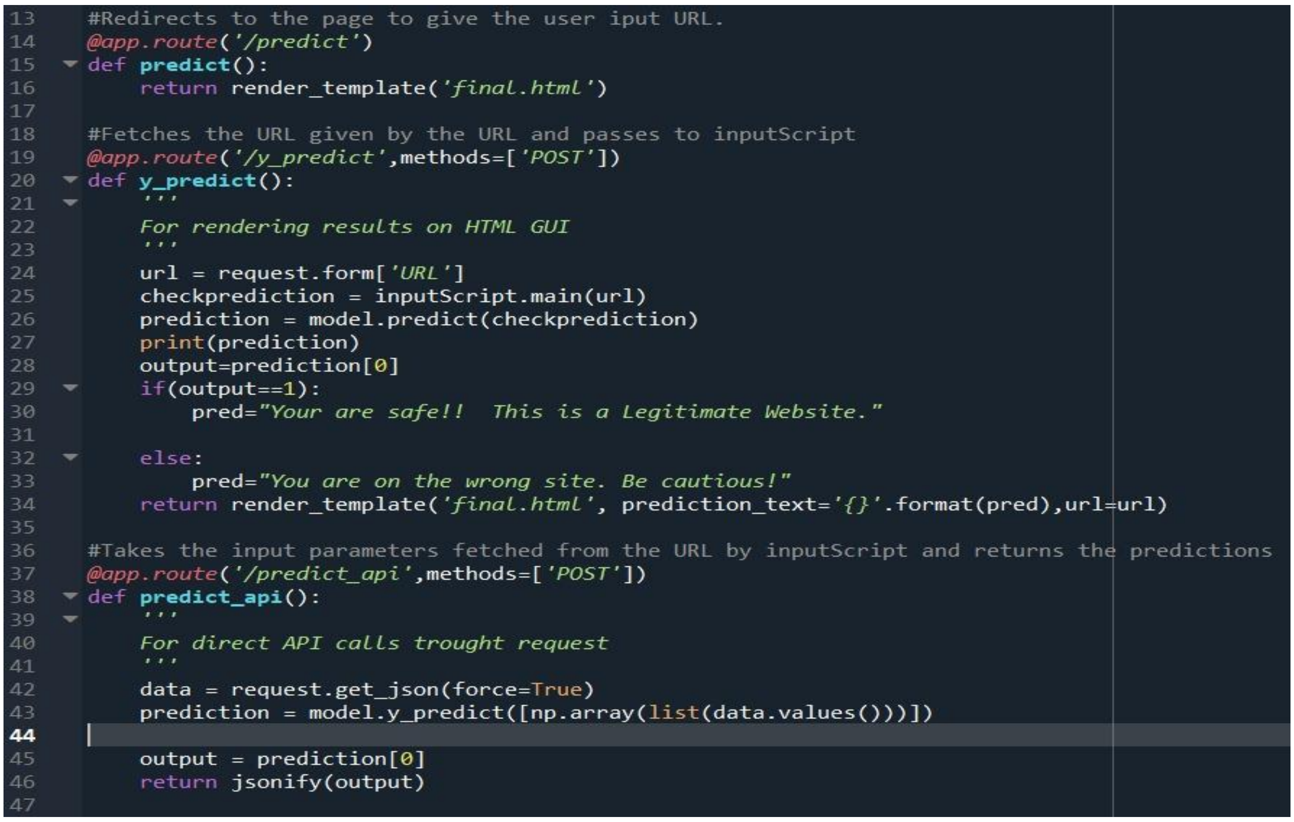


```python
#load model
app = Flask(__name__)
model = pickle.load(open('Phishing_Website.pkl', 'rb'))

```

Fig-4.3.2: Loading the pkl file

Step – 2: Configure app.py to fetch the URL from the UI, process the URL, get the input parameters from the URL and return the prediction.

Input the following commands:

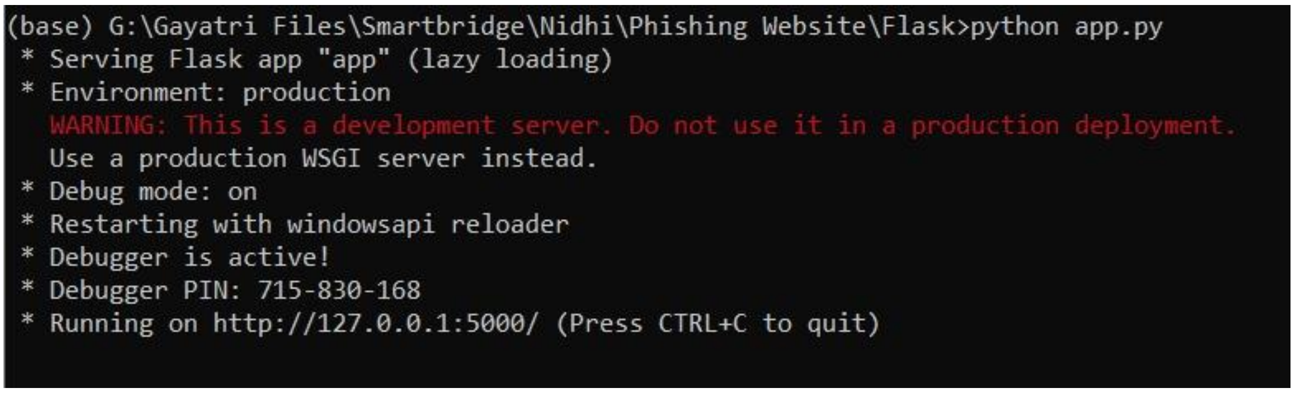


```python
#Redirects to the page to give the user iput URL.
@app.route('/predict')
def predict():
    return render_template('final.html')

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

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

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

    output = prediction[0]
    return jsonify(output)
```

Fig-4.3.3: Fetching the URL

Build An HTML Page:
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. The HTML pages are put under the templates folder and any style sheets if present is kept in the static folder.

Execute And Test Your Model:
Now we execute the model using Anaconda Prompt
Execute the python code by giving the command python app.py in anaconda prompt as shown below

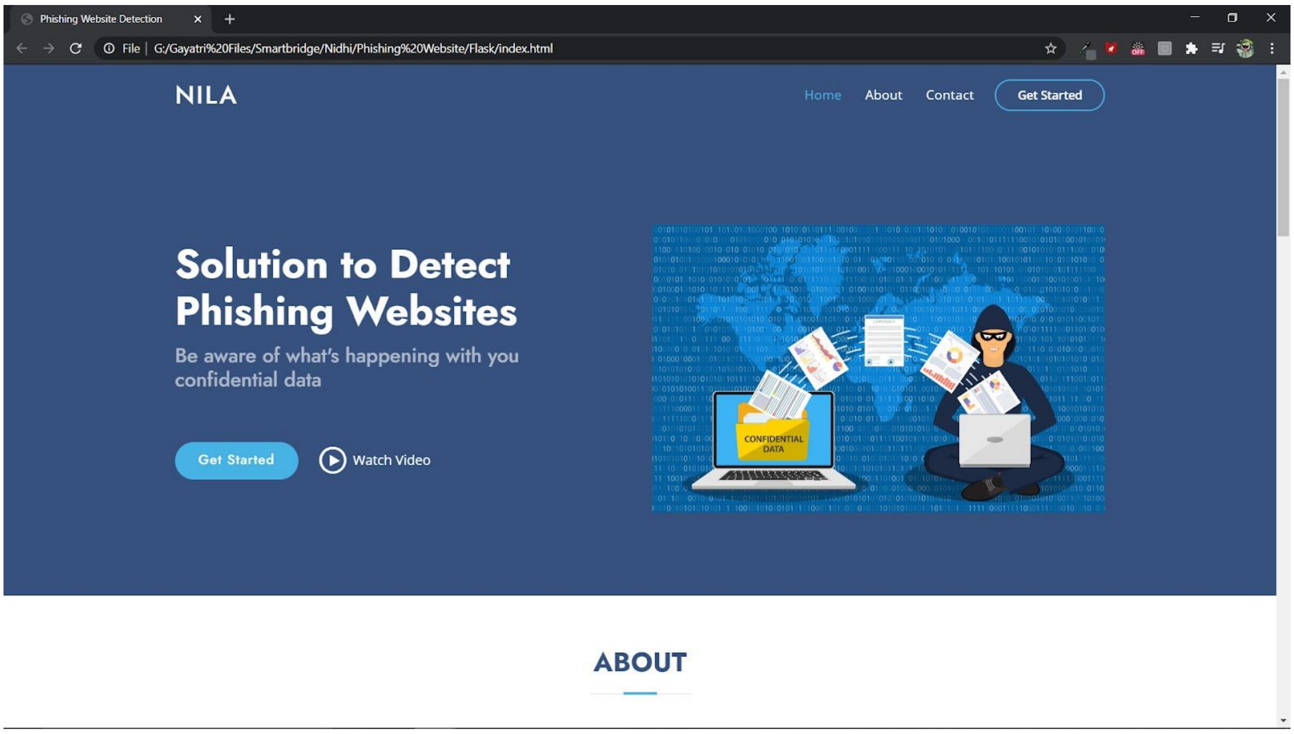


```
(base) G:\Gayatri Files\Smartbridge\Nidhi\Phishing Website\Flask>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 715-830-168
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig-4.3.4: Generating the URL

Now, while the app is running, open index.html present in project folder. Scroll the webpage down to find the buttons to test the application.

This is the home page of the web application(index.html)


Phishing Website Detection

File | G:/Gayatri%20Files/Smartbridge/Nidhi/Phishing%20Website/Flask/index.html

NILA

Home About Contact Get Started

Solution to Detect Phishing Websites

Be aware of what's happening with you confidential data

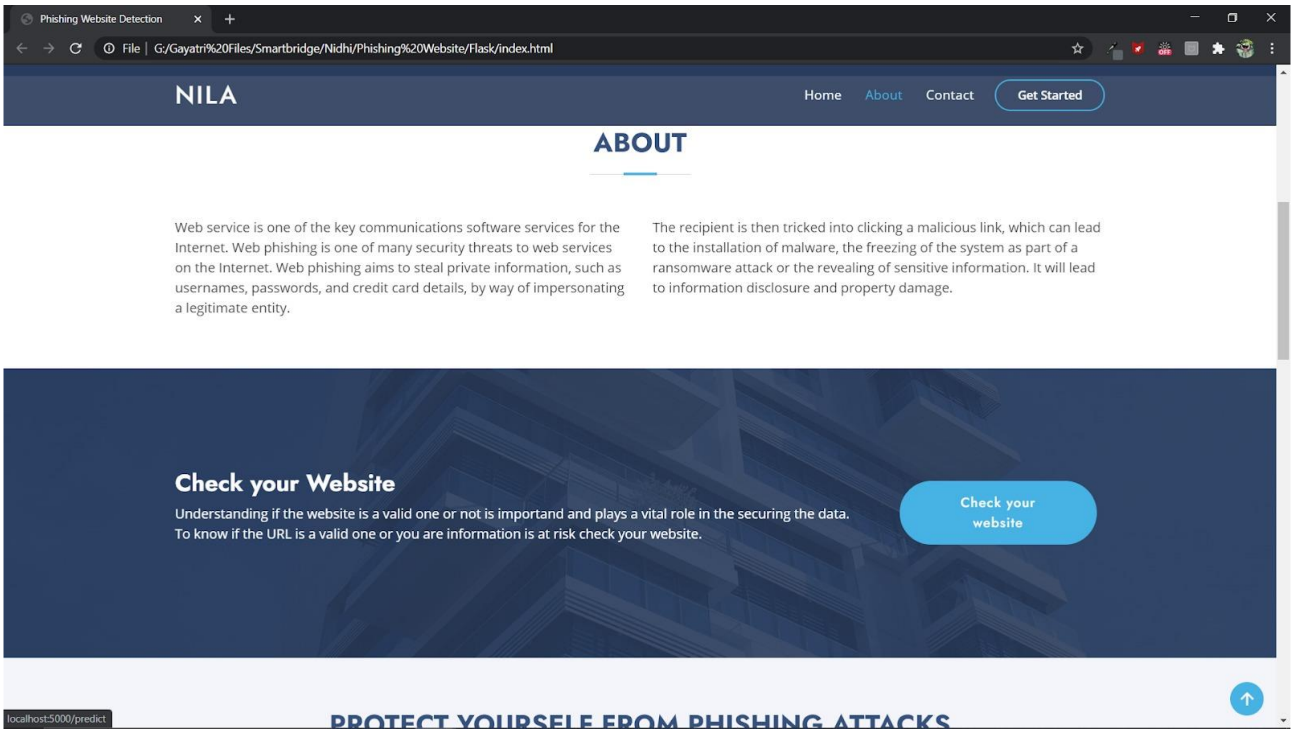
Get Started Watch Video

ABOUT

Fig-4.3.5: Homepage

Testing The Model:

About the project section which gives insights about the project.


Phishing Website Detection

File | G:/Gayatri%20Files/Smartbridge/Nidhi/Phishing%20Website/Flask/index.html

NILA

Home About Contact Get Started

ABOUT

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

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

Check your Website

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

Check your website

PROTECT YOURSELF FROM PHISHING ATTACKS

localhost:5000/predict

Fig-4.3.6: Testing the Model

When clicked on “Check your website” button, the user will be redirected to the below page where user can specify the URL.

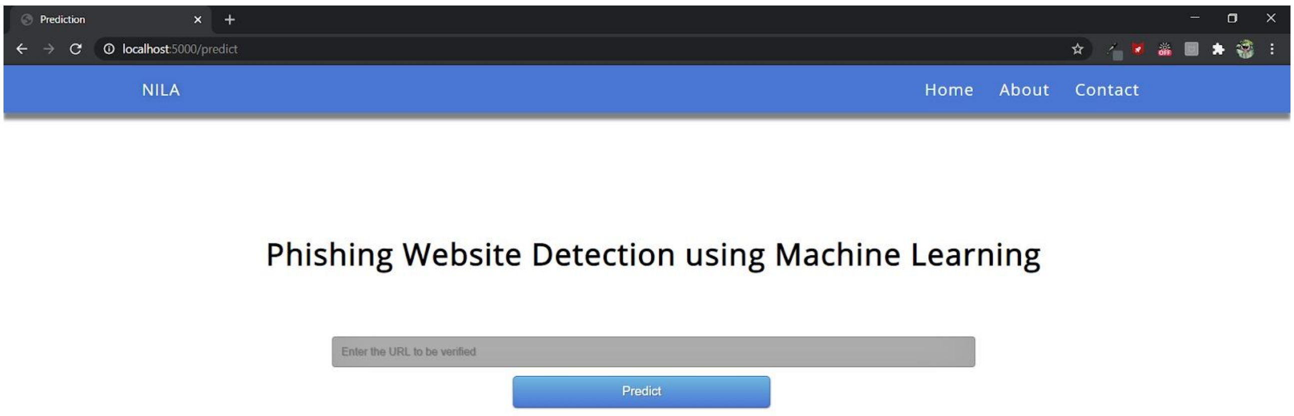
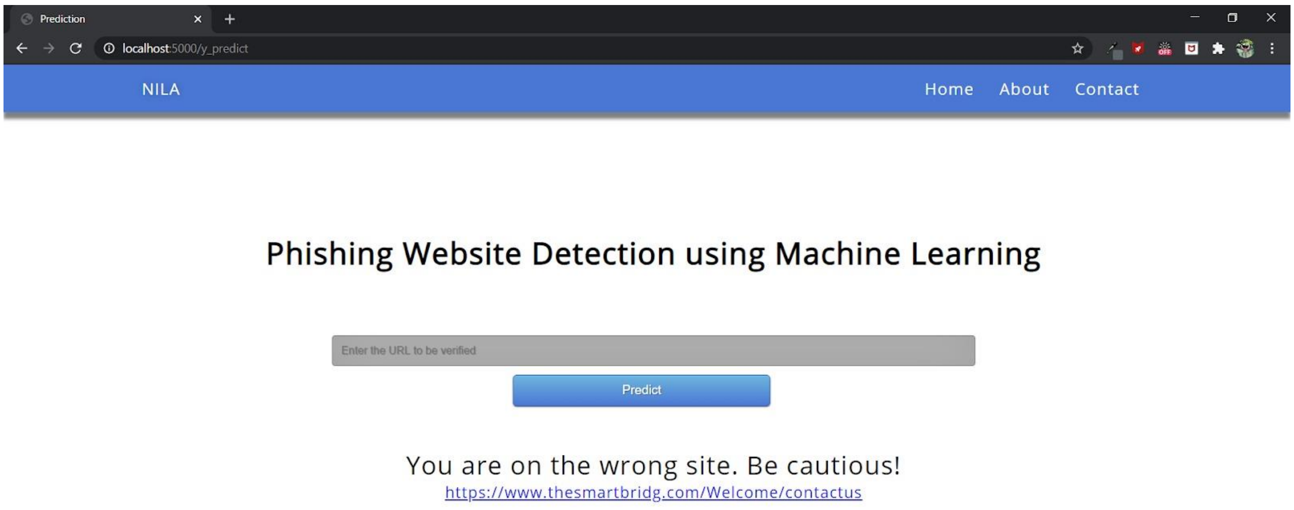

Prediction
localhost:5000/predict
NILA
Home
About
Contact
Phishing Website Detection using Machine Learning
Enter the URL to be verified
Predict

Fig-4.3.7: Phishing Website Detection

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

Here, we will try to specify the same link given above by altering the spelling of the domain name. It validates with the domain name and if not found, It warns about the risk of phishing.

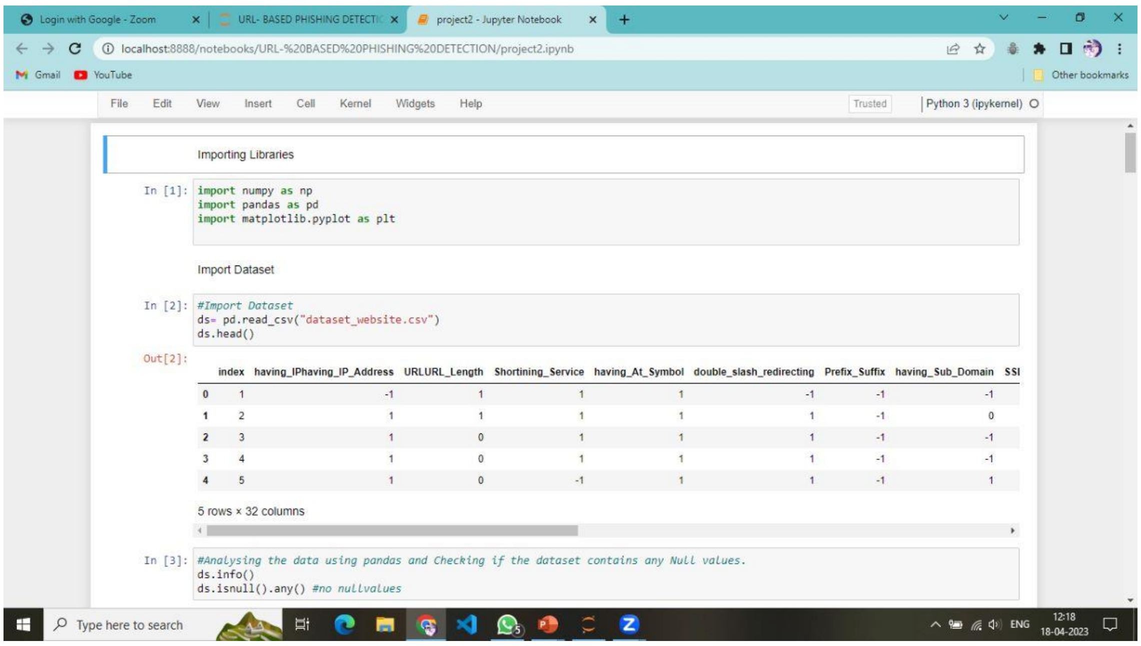

Prediction
localhost:5000/y_predict
NILA
Home
About
Contact
Phishing Website Detection using Machine Learning
Enter the URL to be verified
Predict

You are on the wrong site. Be cautious!
https://www.thesmartbridg.com/Welcome/contactus

Fig-4.3.8: Generating the output

4.4 Appendix

A. Screenshots



Importing Libraries

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

Import Dataset

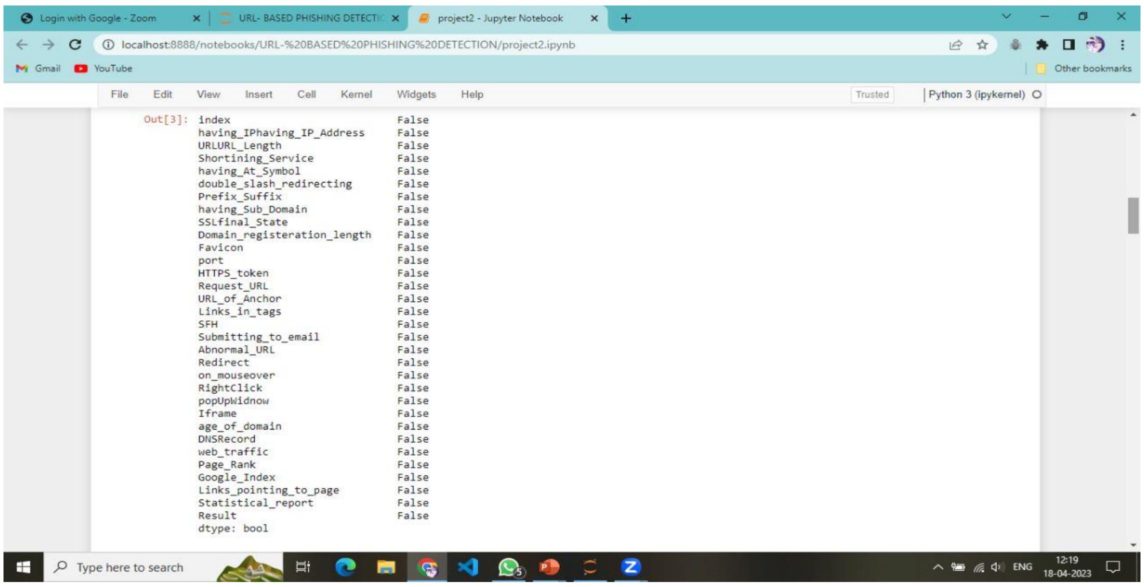
```
In [2]: #Import Dataset
        ds= pd.read_csv("dataset_website.csv")
        ds.head()
```

Out[2]:

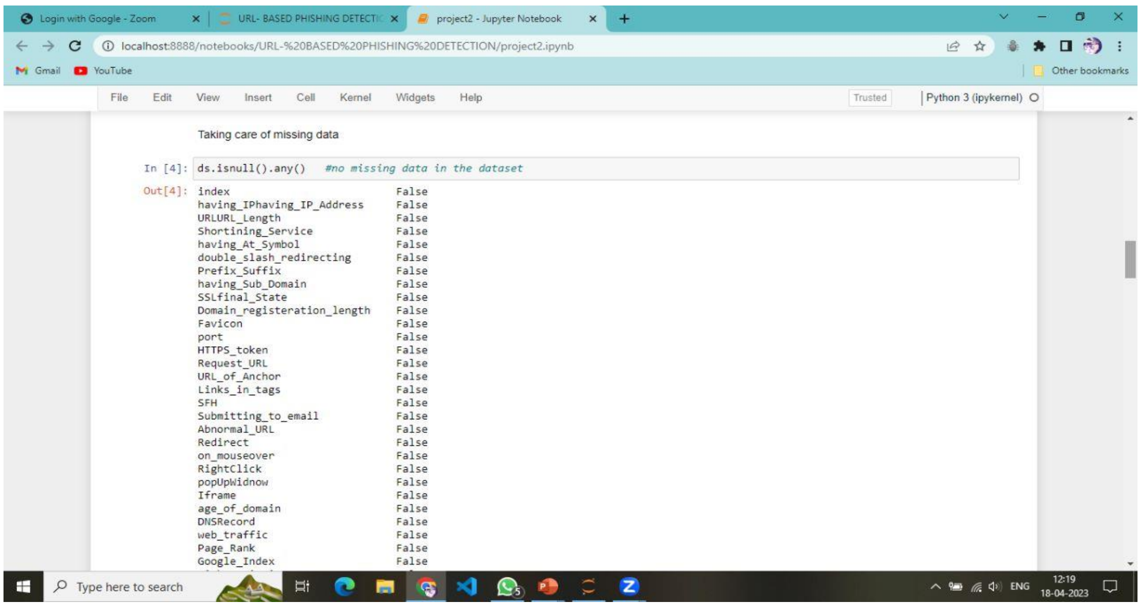
	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSl
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 32 columns

```
In [3]: #Analysing the data using pandas and Checking if the dataset contains any Null values.
        ds.info()
        ds.isnull().any() #no nullvalues
```

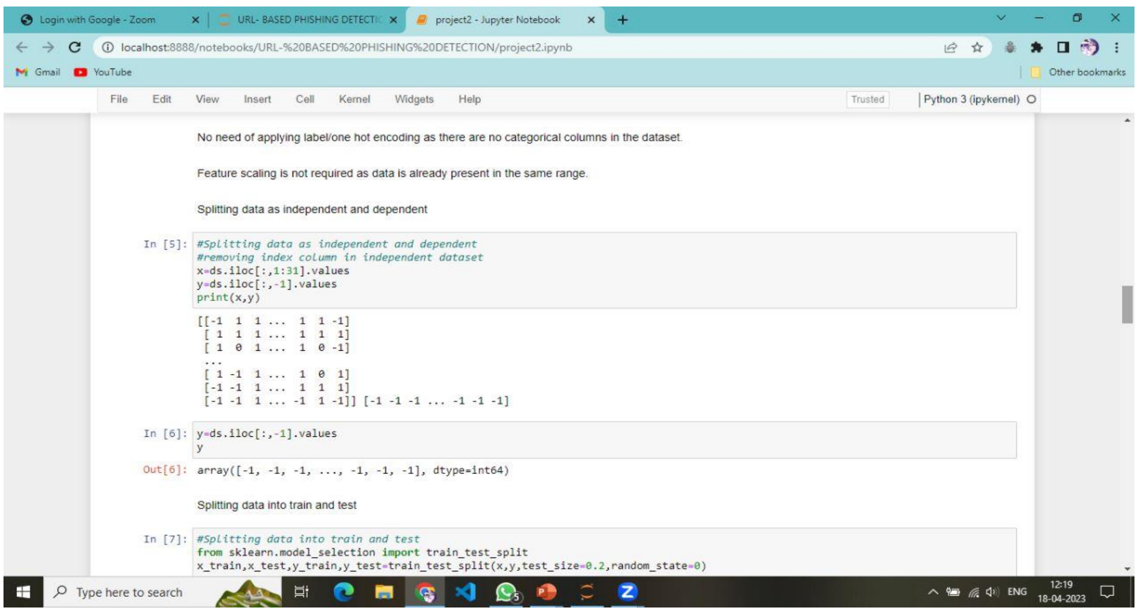


```
Out[3]: index                           False
        having_IPhaving_IP_Address      False
        URLURL_Length                   False
        Shortining_Service              False
        having_At_Symbol                False
        double_slash_redirecting        False
        Prefix_Suffix                   False
        having_Sub_Domain               False
        SSLfinal_State                  False
        Domain_registeration_length     False
        Favicon                         False
        port                            False
        HTTPS_token                     False
        Request_URL                     False
        URL_of_Anchor                   False
        Links_in_tags                   False
        SFH                             False
        Submitting_to_email             False
        Abnormal_URL                    False
        Redirect                        False
        on_mouseover                    False
        RightClick                      False
        popUpWidnow                     False
        Iframe                          False
        age_of_domain                   False
        DNSRecord                       False
        web_traffic                     False
        Page_Rank                       False
        Google_Index                    False
        Links_pointing_to_page          False
        Statistical_report              False
        Result                          False
        dtype: bool
```



Taking care of missing data

```
In [4]: ds.isnull().any()    #no missing data in the dataset

Out[4]: index                          False
        having_IPhaving_IP_Address     False
        URLURL_Length                  False
        Shortining_Service             False
        having_At_Symbol               False
        double_slash_redirecting       False
        Prefix_Suffix                  False
        having_Sub_Domain              False
        SSLfinal_State                 False
        Domain_registeration_length    False
        Favicon                        False
        port                           False
        HTTPS_token                    False
        Request_URL                    False
        URL_of_Anchor                  False
        Links_in_tags                  False
        SFH                            False
        Submitting_to_email            False
        Abnormal_URL                   False
        Redirect                       False
        on_mouseover                   False
        RightClick                     False
        popUpWidnow                    False
        Iframe                         False
        age_of_domain                  False
        DNSRecord                      False
        web_traffic                    False
        Page_Rank                      False
        Google_Index                   False
```



No need of applying label/one hot encoding as there are no categorical columns in the dataset.

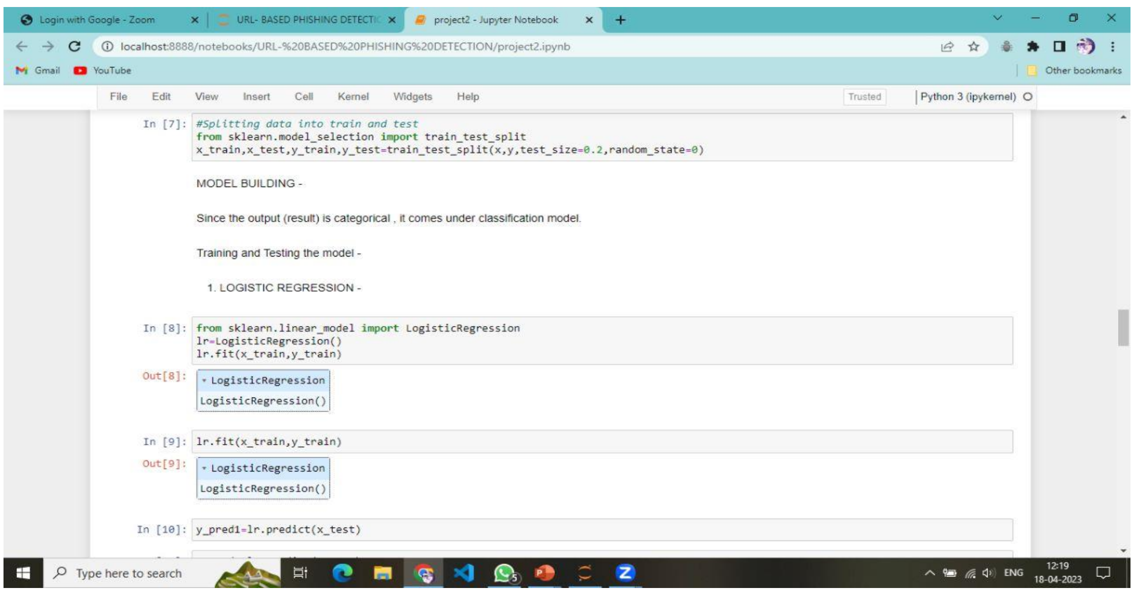
Feature scaling is not required as data is already present in the same range.

Splitting data as independent and dependent

```
In [5]: #Splitting data as independent and dependent
        #removing index column in independent dataset
        x=ds.iloc[:,1:31].values
        y=ds.iloc[:,-1].values
        print(x,y)

        [[-1  1  1 ...  1  1 -1]
         [ 1  1  1 ...  1  1  1]
         [ 1  0  1 ...  1  0 -1]
         ...
         [ 1 -1  1 ...  1  0  1]
         [-1 -1  1 ...  1  1  1]
         [-1 -1  1 ... -1  1 -1]] [-1 -1 -1 ... -1 -1 -1]

In [6]: y=ds.iloc[:,-1].values
        y

Out[6]: array([-1, -1, -1, ..., -1, -1, -1], dtype=int64)
```

Splitting data into train and test

```
In [7]: #Splitting data into train and test
        from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```



```
In [7]: #Splitting data into train and test
        from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

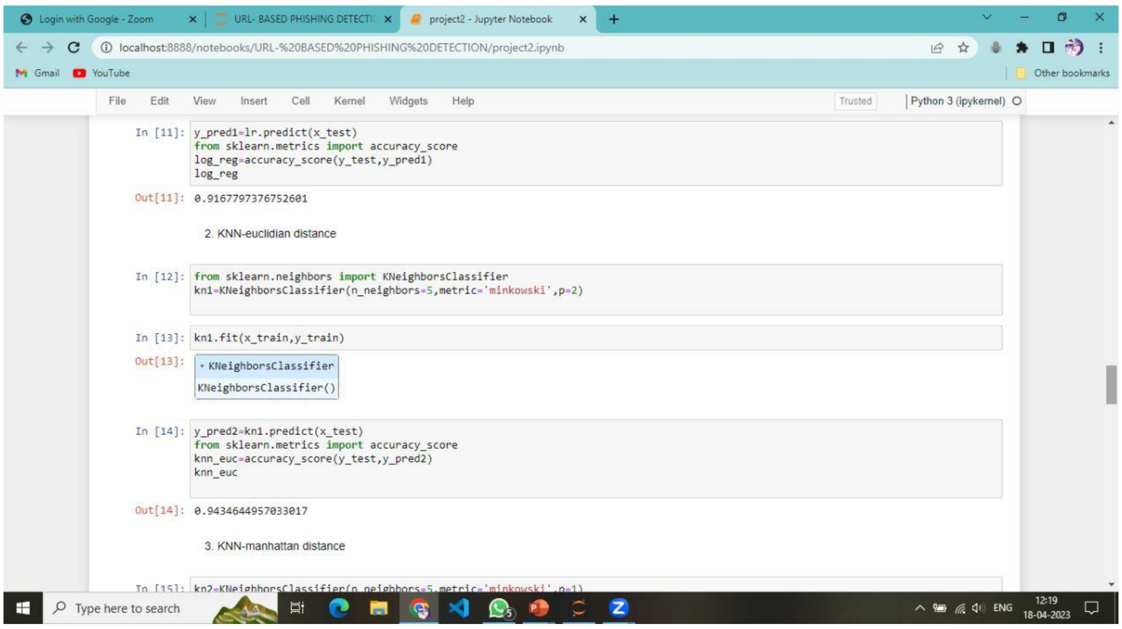
MODEL BUILDING -

Since the output (result) is categorical , it comes under classification model.

Training and Testing the model -

1. LOGISTIC REGRESSION -

```
In [8]: from sklearn.linear_model import LogisticRegression
        lr=LogisticRegression()
        lr.fit(x_train,y_train)

Out[8]: ▸ LogisticRegression
        LogisticRegression()

In [9]: lr.fit(x_train,y_train)

Out[9]: ▸ LogisticRegression
        LogisticRegression()

In [10]: y_pred1=lr.predict(x_test)
```



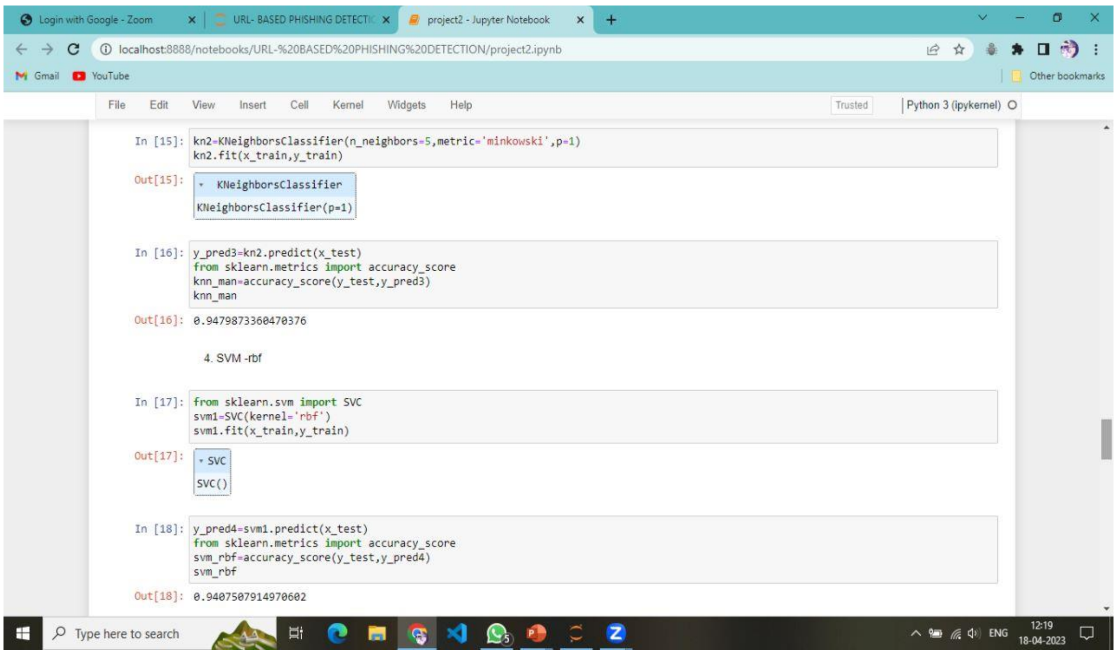
```
In [11]: y_pred1=lr.predict(x_test)
         from sklearn.metrics import accuracy_score
         log_reg=accuracy_score(y_test,y_pred1)
         log_reg

Out[11]: 0.9167797376752601
```

2. KNN-euclidian distance

```
In [12]: from sklearn.neighbors import KNeighborsClassifier
         kn1=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)

In [13]: kn1.fit(x_train,y_train)

Out[13]: ▸ KNeighborsClassifier
         KNeighborsClassifier()

In [14]: y_pred2=kn1.predict(x_test)
         from sklearn.metrics import accuracy_score
         knn_euc=accuracy_score(y_test,y_pred2)
         knn_euc

Out[14]: 0.9434644957033017
```

3. KNN-manhattan distance

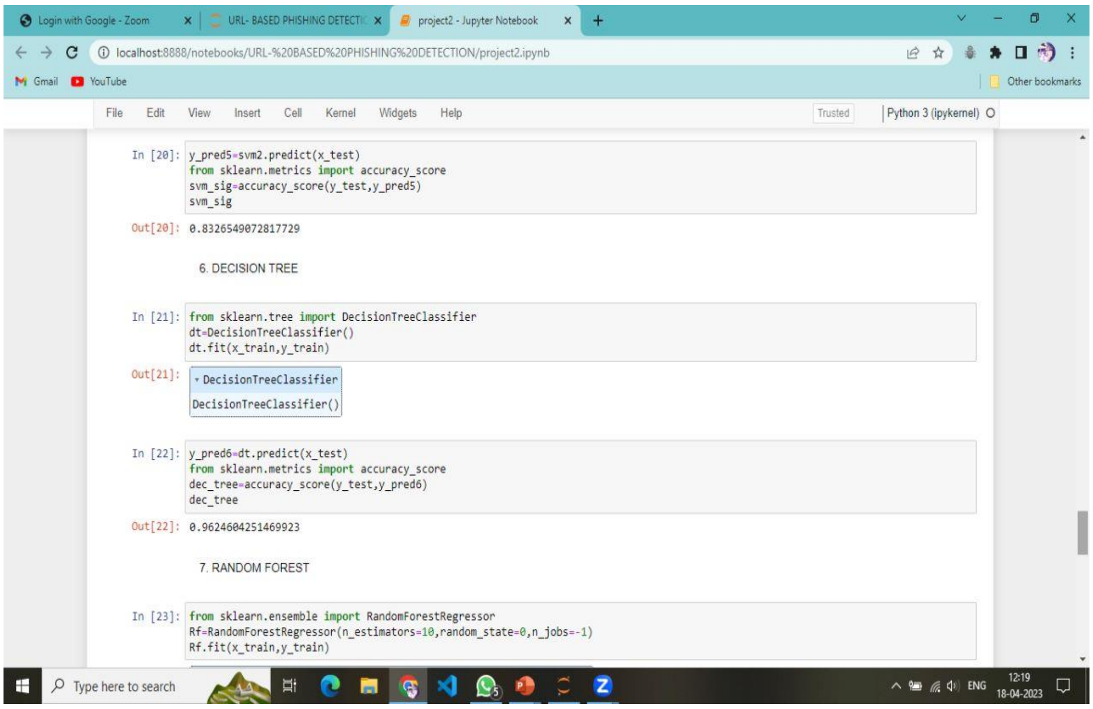
```
In [15]: kn2=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=1)
```



```
In [15]: kn2=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=1)
         kn2.fit(x_train,y_train)

Out[15]: ▸ KNeighborsClassifier
         KNeighborsClassifier(p=1)

In [16]: y_pred3=kn2.predict(x_test)
         from sklearn.metrics import accuracy_score
         knn_man=accuracy_score(y_test,y_pred3)
         knn_man

Out[16]: 0.9479873360470376
```

4. SVM -rbf

```
In [17]: from sklearn.svm import SVC
         svm1=SVC(kernel='rbf')
         svm1.fit(x_train,y_train)

Out[17]: ▸ SVC
         SVC()

In [18]: y_pred4=svm1.predict(x_test)
         from sklearn.metrics import accuracy_score
         svm_rbf=accuracy_score(y_test,y_pred4)
         svm_rbf

Out[18]: 0.9407507914970602
```



6. DECISION TREE

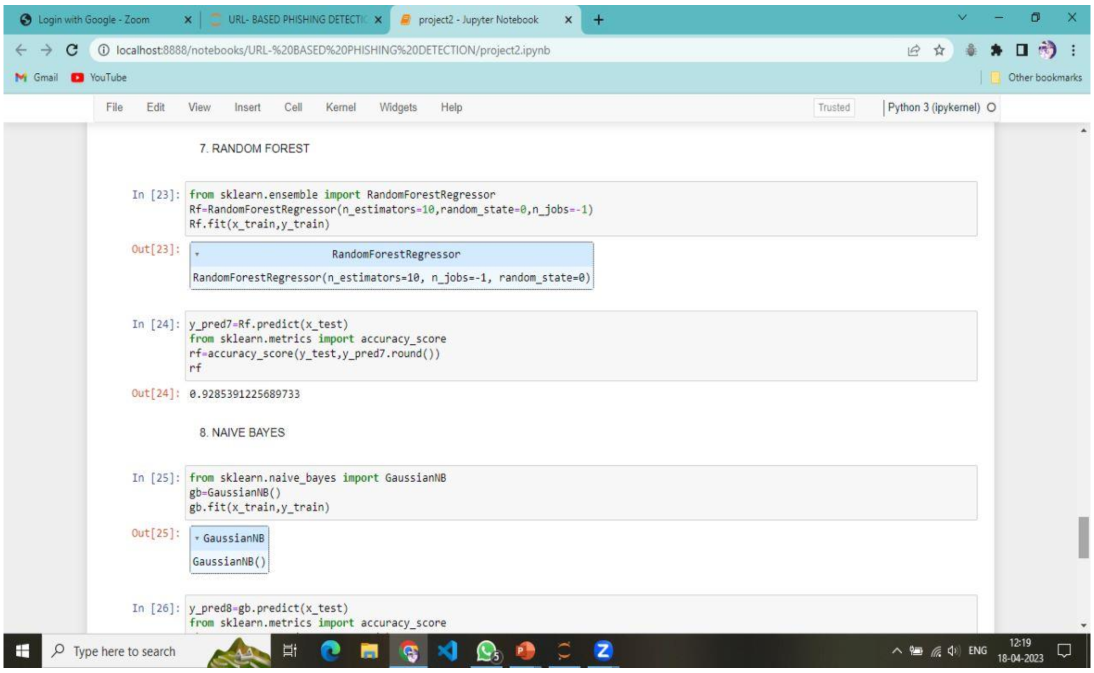
```
In [20]: y_pred5=svm2.predict(x_test)
         from sklearn.metrics import accuracy_score
         svm_sig=accuracy_score(y_test,y_pred5)
         svm_sig
Out[20]: 0.8326549072817729
```

```
In [21]: from sklearn.tree import DecisionTreeClassifier
         dt=DecisionTreeClassifier()
         dt.fit(x_train,y_train)
Out[21]: ▸ DecisionTreeClassifier
         DecisionTreeClassifier()
```

```
In [22]: y_pred6=dt.predict(x_test)
         from sklearn.metrics import accuracy_score
         dec_tree=accuracy_score(y_test,y_pred6)
         dec_tree
Out[22]: 0.9624604251469923
```

7. RANDOM FOREST

```
In [23]: from sklearn.ensemble import RandomForestRegressor
         Rf=RandomForestRegressor(n_estimators=10,random_state=0,n_jobs=-1)
         Rf.fit(x_train,y_train)
```



7. RANDOM FOREST

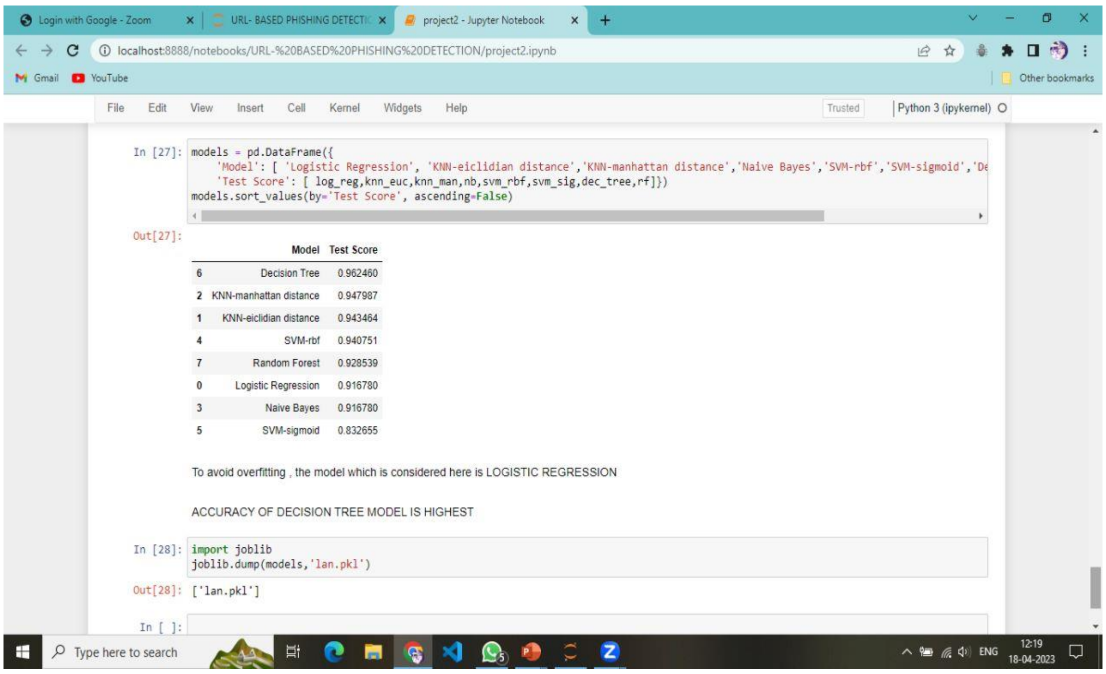
```
In [23]: from sklearn.ensemble import RandomForestRegressor
         Rf=RandomForestRegressor(n_estimators=10,random_state=0,n_jobs=-1)
         Rf.fit(x_train,y_train)
Out[23]: ▸                 RandomForestRegressor
         RandomForestRegressor(n_estimators=10, n_jobs=-1, random_state=0)
```

```
In [24]: y_pred7=Rf.predict(x_test)
         from sklearn.metrics import accuracy_score
         rf=accuracy_score(y_test,y_pred7.round())
         rf
Out[24]: 0.9285391225689733
```

8. NAIVE BAYES

```
In [25]: from sklearn.naive_bayes import GaussianNB
         gb=GaussianNB()
         gb.fit(x_train,y_train)
Out[25]: ▸ GaussianNB
         GaussianNB()
```

```
In [26]: y_pred8=gb.predict(x_test)
         from sklearn.metrics import accuracy_score
```



```
In [27]: models = pd.DataFrame({
             'Model': [ 'Logistic Regression', 'KNN-eiclidian distance','KNN-manhattan distance','Naive Bayes','SVM-rbf','SVM-sigmoid','De
             'Test Score': [ log_reg,knn_euc,knn_man,nb,svm_rbf,svm_sig,dec_tree,rf]})
         models.sort_values(by='Test Score', ascending=False)

Out[27]:
                    Model  Test Score
   6        Decision Tree    0.962460
   2 KNN-manhattan distance  0.947987
   1   KNN-eiclidian distance  0.943464
   4              SVM-rbf    0.940751
   7        Random Forest    0.928539
   0  Logistic Regression    0.916780
   3          Naive Bayes    0.916780
   5          SVM-sigmoid    0.832655
```

To avoid overfitting , the model which is considered here is LOGISTIC REGRESSION

ACCURACY OF DECISION TREE MODEL IS HIGHEST

```
In [28]: import joblib
         joblib.dump(models,'lan.pkl')

Out[28]: ['lan.pkl']

In [ ]:
```

B. Source code

Python code (app.py)

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


#load model
app = Flask(_name_)
model = pickle.load(open('lan.pkl', 'rb'))


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

@app.route('/predict')
def predict():
    return render_template('final.html')
```

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

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

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

    output = prediction[0]
    return jsonify(output)

if _name_ == "_main_":
    app.run(debug=True)

if _name_ == '_main_':
		app.run(host='0.0.0.0', debug=True)
```

Html code :

index.html

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

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

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



  <!-- Google Fonts -->
  <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,70
0,700i|Jost:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,
500i,600,600i,700,700i" rel="stylesheet">

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


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

```
                                    <link href="{{ url_for('static', filename='/vendor/aos/aos.css') }}"
rel="stylesheet"-->

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


</head>

<body>

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

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

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

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

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

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

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

    <div class="container">
      <div class="row">
        <div class="col-lg-6 d-flex flex-column justify-content-center pt-4 pt-lg-0 order-2 order-
lg-1" data-aos="fade-up" data-aos-delay="200">
          <h1>Solution to Detect Phishing Websites         </h1>
          <h2>Be aware of what's happening with you confidential data</h2>
          <div class="d-lg-flex">
            <a href="/predict" class="btn-get-started scrollto">Get Started</a>
            <a href="https://www.youtube.com/watch?v=jDDaplaOz7Q" class="venobox btn-
watch-video" data-vbtype="video" data-autoplay="true"> Watch Video <i class="icofont-
play-alt-2"></i></a>
          </div>
        </div>
        <div class="col-lg-6 order-1 order-lg-2 hero-img" data-aos="zoom-in" data-aos-
delay="200">
```

```
          <img src="https://www.technologyvisionaries.com/wp-
content/uploads/2020/01/bigstock-Data-Phishing-Hacker-Attack-t-319270852-scaled.jpg"
class="img-fluid animated" alt="">
        </div>
      </div>
    </div>

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

  <main id="main">


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

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

        <div class="row content">
          <div class="col-lg-6">
            <p>
              Web service is one of the key communications software services for the Internet.
Web phishing is one of many security threats to web services on the Internet.  Web
phishing aims to steal private information, such as usernames, passwords, and credit card
details, by way of impersonating a legitimate entity.
            </p>

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

          </div>
        </div>

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

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

        <div class="row">
          <div class="col-lg-9 text-center text-lg-left">
            <h3>Check your Website</h3>
            <p>Understanding if the website is a valid one or not is importand and plays a vital
role in the securing the data. To know if the URL is a valid one or you are information is at
```

```
risk check your website. </p>
          </div>
          <div class="col-lg-3 cta-btn-container text-center">
            <a class="cta-btn align-middle" href="http://localhost:5000/predict">Check your
website</a>
          </div>
        </div>

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


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

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

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

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

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

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





        </div>

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







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

        <div class="section-title">
          <h2>Contact</h2>
          <p>Get in contect with us to build many more amazing projects.</p>
        </div>

        <div class="row">

          <div class="col-lg-9 mt-5 mt-lg-0 d-flex align-items-stretch">
            <div class="info">
              <div class="address">
                <i class="icofont-google-map"></i>
                <h4>Location:</h4>
                <p>Nacharam Main Road,Hyderabad</p>
              </div>

              <div class="email">
                <i class="icofont-envelope"></i>
                <h4>Email:</h4>
                <p>info@thesmartbridge.com</p>
              </div>
              <div class="phone">
                <i class="icofont-phone"></i>
                <h4>Call:</h4>
                <p>9122223335</p>
              </div>

                                                    <iframe width="100%" height="370" frameborder="0"
allowfullscreen="" style="border:0"
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d15227.078752666794!
2d78.545269!3d17.422837!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3bcb995da7380f6b%3
A0x608ed5e4b34e713b!2sNarmada+Arcade%2C+Nacharam+Mallapur+Rd%2C+Snehap
uri+Colony%2C+Tarnaka%2C+Hyderabad%2C+Telangana+500076%2C+India!5e0!3m2!
1sen!2sin!4v1452762479907" frameborder="0" style="border:0; width: 100%; height:
290px;" allowfullscreen></iframe>

            </div>

          </div>

          <div class="col-lg-7 mt-5 mt-lg-0 d-flex align-items-stretch">


          </div>

        </div>

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

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

  <!-- ======= Footer ======= -->
  <footer id="footer">



    <div class="footer-top">
      <div class="container">
        <div class="row">


        </div>
      </div>
    </div>

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

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

  <a href="#" class="back-to-top"><i class="ri-arrow-up-line"></i></a>
  <div id="preloader"></div>
  <!-- Vendor JS Files -->
  <script src="{{ url_for('static', filename='assets/vendor/jquery/jquery.min.js')}}"></script>
  <script src="{{url_for('static',
filename='assets/vendor/bootstrap/js/bootstrap.bundle.min.js')}}"></script>
  <script src="{{ url_for('static',
filename='assets/vendor/jquery.easing/jquery.easing.min.js')}}"></script>
  <script src="{{ url_for('static', filename='assets/vendor/php-email-
form/validate.js')}}"></script>
  <script src="{{ url_for('static',
filename='assets/vendor/waypoints/jquery.waypoints.min.js')}}"></script>
  <script src="{{ url_for('static', filename='assets/vendor/isotope-
layout/isotope.pkgd.min.js')}}"></script>
  <script src="{{ url_for('static',
filename='assets/vendor/venobox/venobox.min.js')}}"></script>
  <script src="{{ url_for('static',
filename='assets/vendor/owl.carousel/owl.carousel.min.js')}}"></script>
  <script src="{{ url_for('static', filename='assets/vendor/aos/aos.js')}}"></script>

  <!-- Template Main JS File -->
  <script src="{{ url_for('static', filename='assets/js/main.js')}}"></script>


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

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

</body>

</html>
```

final.html

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

```
rel='stylesheet' type='text/css'>
                                <link href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
                                <link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
                                <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
                                <link rel="stylesheet" href="{{ url_for('static',
filename='css/final1.css') }}">
                                <!--link rel="stylesheet" href="G:\Gayatri
Files\Smartbridge\Nidhi\Phishing Website\static\css\style1.css"-->

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

<body>
<div class="header">
<div>Web Phishing Detection</div>
                                <ul>


                                 <li><a href="/#contact">Contact</a></li>
                                 <li><a href="/#about">About</a></li>
                                 <li><a href="/">Home</a></li>
                                </ul>
</div>

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

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

</html>
```

CHAPTER – 5
SUMMARY AND CONCLUSIONS

5.1 Summary

Phishing attack are continually advancing and the digital world is hit by new kinds of assaults frequently. Consequently a specific location approach or calculation can't be labeled as the best one giving precise outcomes. Through the writing study, We discovered that Support Vector Machine gives better outcomes in many situations. However at that point the exhibition of every calculation differs relying upon the dataset utilized, train-test split proportion, highlight determination strategies applied, and so forth Scientists like to make AI models that perform phishing location with the best incentive for assessment boundaries and least preparing time. Subsequently, our future works center around working on these parts of phishing identification.

5.2 conclusion

Phishing detection is currently an area of incredible interest among specialists because of its importance in ensuring the protection and giving security. Numerous techniques perform phishing location by characterization sites utilizing prepared AI models. In this paper, we depicted our precise study of existing URL-based phishing identification procedures from various perspectives. Albeit past overview papers exist, they by and large spotlight on in general phishing location methods, while we zeroed in on itemized URL-based discovery concerning highlights. Right off the bat, we audited the writing on by and large phishing identification plans. Second, we examined the design of URL-based phishing, and ordinarily utilized calculations and highlights. Third, normal information sources were recorded, and near assessment results and grids were displayed for a superior study. At long last, we closed with our idea to continue with the Support Vector Algorithm for more successful phishing URL identification in our venture.

References

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.