1. INTRODUCTION

1.1 ABSTRACT

With raising in-depth amalgamation of the Internet and social life, the Internet is looking differently at how people are learning and working, meanwhile opening us to growing serious security attacks. The ways to recognize various network threats, specifically attacks not seen before, is a primary issue that needs to be looked into immediately. The aim of phishing site URLs is to collect private information like user's identity, passwords and online money related exchanges. Phishers use the sites which are visibly and semantically like those of authentic websites. Since the majority of the clients go online to get to the administrations given by the government and money related organizations, there has been a vital increment in phishing threats and attacks since some years.

As technology is growing, phishing methods have started to progress briskly and this should be avoided by making use of anti-phishing techniques to detect phishing. Machine learning is an authoritative tool that can be used to aim against phishing assaults. There are several methods or approaches to identify phishing websites. The machine learning approaches to detect phishing websites have been proposed earlier and have been implemented. The central aim of this project is to implement the system with high efficiency, accuracy and cost effectively. That has been achieved. The project is implemented using 5 machine learning supervised classification models. The five classification models are K-Nearest Neighbor, Decision Tree, Logistic Regression, Random Forest classifier and Gradient Boosting Classifier. It was established that the Gradient Boosting Classifier provides best accuracy for the selected dataset and gives an accuracy score of 97.4%.

Quick response (QR) code gained popularity and has been adapted for various applications such as a pointer to digital information and authentication. While the code gives convenience as a physical pointer to the digital world, it can be manipulated to divert the intended destination of the link to a malicious site. Thus, QR codes can be easily exploited by phishers to launch phishing attacks. So, by uploading that suspicious QR to model it identifies whether there is any link in it and detects the link for phishing or legitimate.

1.2 PROBLEM STATEMENT

The problem is derived after making a thorough observation and study about the method of classification of phishing websites that makes use of machine learning techniques. We must design a system that should allow us to:

- Accurately and efficiently classify the websites into legitimate or phishing.
- Time consumed for detection should be less and should be cost effective.

As these URL phishing attacks are also now taken as a new form of QR code phishing by scanning that QR we redirect to some website and attack happens. To overcome this attack also, we designed to detect that QR code URL.

1.3 SCOPE

The focus of the project is on machine learning (ML) methods for network analysis of intrusion detection especially phishing websites attack.

1.4 CHALLENGES

The challenges faced during the project are as follows:

- Finding the appropriate dataset.
- Feature extraction required the study of various modules and understanding each module and getting the expected outcome from it.

2. MODULES

2.1 Module Description

A Module Description provides detailed information about the module and its supported components, which is accessible in different manners.

The modules are,

- Input URL
- Upload QR
- Detection of URL

2.1.1 Input URL

The url that you suspect, whether it is legitimate or phishing is given to the model. The sentence that contains the url is given, so that the Algorithm separates the url and checks whether it is legitimate or phishing.

2.1.2 Upload QR

The QR code is decoded by uploading it and the decoded text is parsed to the model and the model checks for malicious links and predicts.

2.1.3 Detection of URL

On the successful detection of the URL, the predictions and accuracy of the prediction is shown as output.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

COMPONENTS	REQUIREMENTS
CPU	**Intel Pentium Dual Core and Higher**
Hard Disk Capacity	**512MB Space required minimum**
RAM	**4GB minimum**

table 1

3.2 SOFTWARE REQUIREMENTS

COMPONENTS	REQUIREMENTS
Python	**Version 3.x**
Operating System	**Windows 10 or above**
Jupyterlab	**Version 5 or above**
Apache	**Version 2.x**

table 2

3.3 REQUIREMENTS PYTHON

Python modules:

S.no	Module	Description
1	**BeautifulSoup**	Beautiful Soup is a Python library that is commonly used for web scraping. It allows you to parse HTML and XML documents, extract useful information, and manipulate the contents of the document.
2	**Decode**	Decode is a Python module that provides a simple way to decode barcodes and QR codes from images and video streams. It is a wrapper around the ZBar library, which is an open-source software suite for reading barcodes from various sources

3	**Flask**	Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.
4	**Ipaddress**	ip address provides the capabilities to create, manipulate and operate on IPv4 and IPv6 addresses and networks.
5	**Matplotlib**	Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, Python/IPython shells, web application servers, and various graphical user interface toolkits.
6	**NumPy**	Numerical Python(NumPy) is used for scientific computing and data analysis. It provides a powerful N-dimensional array object, as well as functions for working with arrays and performing mathematical operations.
7	**Pandas**	Pandas provides high-level data structures and functions designed to make working with structured data fast, easy, and expressive. Some of the key features are Data structures, Data Manipulation, Data Cleaning, Data Visualization, Data Input/Output.
8	**Pickle**	The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.
9	**Requests**	The requests module makes it easy to perform common HTTP operations, such as GET, POST, PUT, DELETE, and more, and allows you to interact with RESTful APIs and other web services.
10	**Scikit-Learn**	Scikit-learn (short for "Scientific Kit for Machine Learning") is a popular open-source machine learning library for Python that provides a wide range of tools for data mining, analysis, and modeling. Scikit-learn provides a high-level API that allows developers to quickly and easily implement common machine learning tasks, such as classification, regression, clustering, and dimensionality reduction.
11	**Urllib**	Urllib is a Python module that provides a simple and standard way to handle various types of URL (Uniform Resource Locator)

		interactions, such as opening URLs, reading from URLs, and submitting data to URLs.
12	**werkzeug**	It is a Python web application library that provides a set of utilities and tools for building web applications. It is a lightweight and flexible library that can be used as a standalone library or with a web framework such as Flask.
13	**Xgboost**	XGBoost (eXtreme Gradient Boosting) is a popular open-source library for gradient boosting that is used for machine learning tasks, particularly for building and training ensemble models.

table 3

3.4 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a determination that depicts the framework's activity abilities and requirements that improve its usefulness. Some of them are as follows:

- Reusability: the same code with limited changes can be used for detecting phishing attacks variants like smishing, vishing, etc.
- Maintainability: The implementation is very basic and includes print statements that makes it easy to debug.
- Usability: The software used is very user friendly and open source. It also runs on any Operating system.
- Scalability: The implementation can include detection of vishing, smishing, etc.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

The existing system has three major phases, that are Parsing, Heuristic Classification of data, and Performance Analysis. All of these phases use various and distinctive methods for data processing to get results that are better.

DISADVANTAGES OF EXISTING SYSTEM

- Time consuming as it involves three phases and each website has to go through the three phases.
- False positive rate is high

4.2 PROPOSED SYSTEM

The objective of the proposed system is to reduce the time consuming and make the system more user friendly, efficient, accurate and fast processing. It also provides QR scanning. Website in the QR is decoded and the URL is parsed to detect whether it is legitimate or not.

ADVANTAGES OF PROPOSED SYSTEM

- More User friendly
- Fast and accurate predictions
- Scans QR for knowing that QR is safe or not

5. SOFTWARE OVERVIEW

5.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge of all major platforms, and can be freely distributed.

Python is used in machine learning for a variety of reasons.

1. It is a very concise and readable language, which makes it easy to develop algorithms.
2. Python has a large number of well-developed libraries, which can be used to develop machine learning algorithms.
3. Python is fast enough for most machine learning tasks.

5.2 JUPYTER

Jupyter is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It supports multiple programming languages and is widely used in scientific computing, data analysis, and machine learning. Jupyter notebooks, the primary type of document used in Jupyter, are files that contain both code and rich text elements, such as equations, images, and interactive widgets. They can be easily shared with others, allowing for collaborative coding and data analysis. Jupyter provides a versatile and powerful tool for data scientists, researchers, and educators.

5.3 HTML

HTML stands for HyperText Markup Language. It is a markup language used for creating web pages and other online content that can be displayed on web browsers. HTML uses tags and attributes to structure and format text, images, links, and other content on a web page. These tags and attributes define the structure and layout of a web page, and provide a way for web developers to create content that can be easily read and understood by web browsers.

HTML is an essential part of web development, and is often used in conjunction with other technologies like CSS (Cascading Style Sheets) and JavaScript to create rich, interactive web pages and applications. Web developers use HTML to create the basic structure and content of a web page, while CSS is used to define the presentation and layout of the content, and JavaScript is used to add interactivity and dynamic behavior to the page.

5.4 CSS

CSS (Cascading Style Sheets) is a language used for describing the presentation of a document written in HTML or XML. CSS describes how HTML elements should be displayed on screen, paper, or in other media. It controls the layout, fonts, colors, and other visual aspects of a web page, allowing web developers to separate content from presentation.

5.5 JAVASCRIPT

JavaScript is a programming language that is used primarily for creating interactive and dynamic web pages. It allows developers to add functionality to HTML pages, such as creating animations, validating form data, and responding to user events such as mouse clicks or keyboard input. JavaScript can be used to manipulate the content of a web page dynamically, allowing for a more engaging and responsive user experience. It is often used in conjunction with HTML and CSS to create interactive web applications.

6. SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE DESIGN

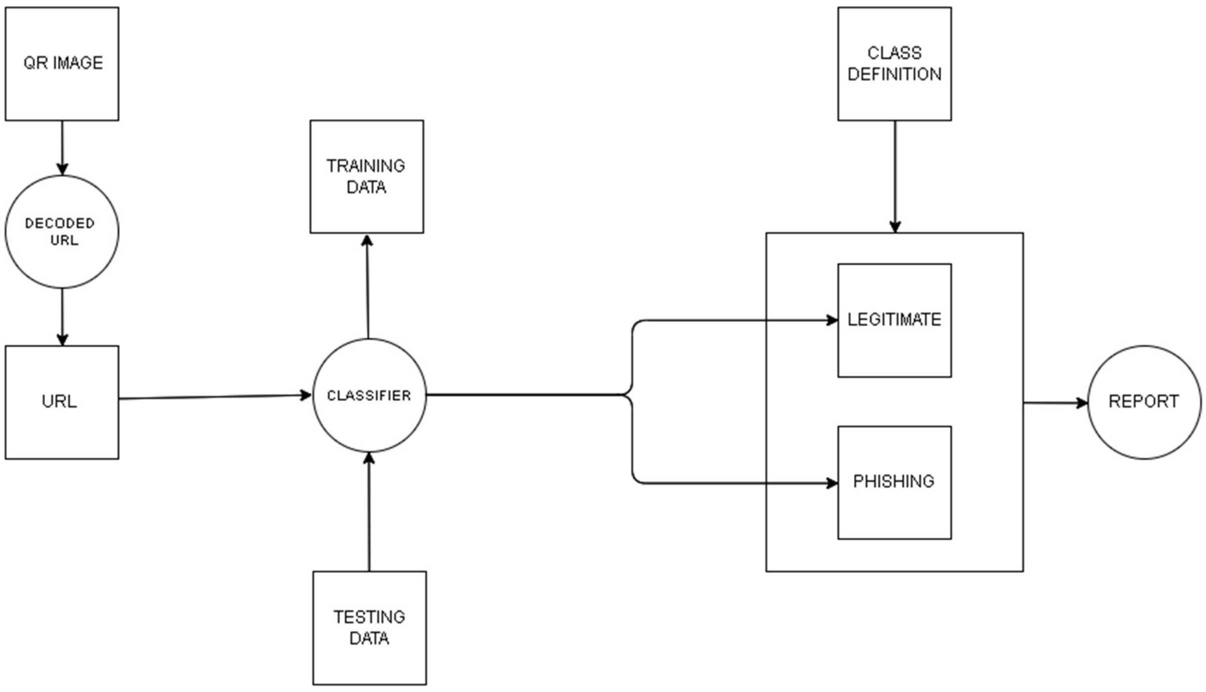


figure.1

The architecture of the system is as shown in figure.1; the URLs to be classified as legitimate or phishing is fed as input to the appropriate classifier. Then the classifier that is being trained to classify URLs as phishing or legitimate from the training dataset uses the pattern it recognized to classify the newly fed input.

The features such as IP address, URL length, domain, having favicon, etc. are extracted from the URL and a list of its values is generated. The list is fed to the classifiers such as KNN, Logistic regression, Decision tree, Random Forest classifier and Gradient Boosting classifier. These models' performance is then evaluated and an accuracy score is generated. The trained classifier using the generated list predicts if the URL is legitimate or phishing. The list contains values 1, 0 and -1 if the features exist, not applicable and if the features don't exist respectively. There are 30 features being considered in this project.

6.2 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. The use cases are represented by either circles or ellipses.

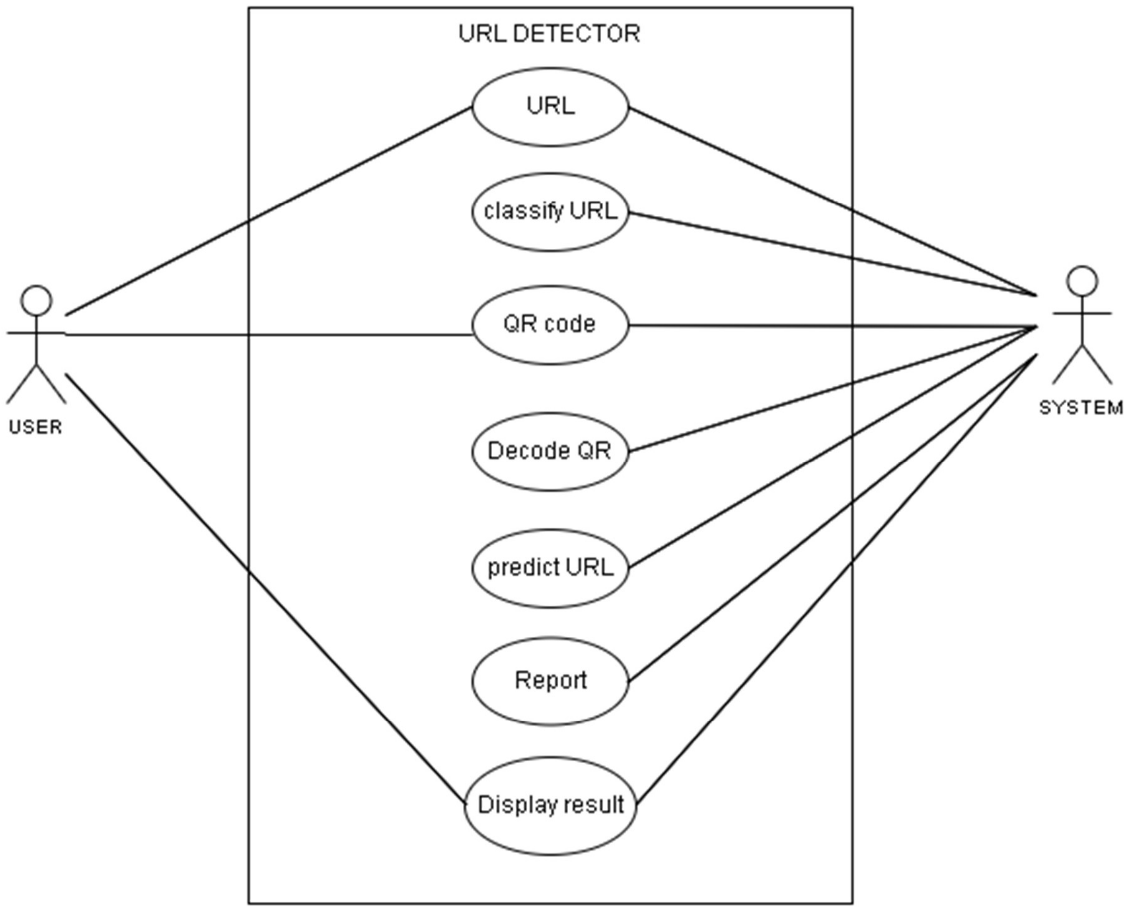


Figure.2

6.3 DATA FLOW DIAGRAM

DFDs are used to graphically depict the data flow in a system. It explains the processes involved in a system from the input to the report generation. It shows all possible paths from one entity to another of a system. The detail of a data flow diagram can be represented in three different levels that are numbered 0, 1 and 2.

6.3.1 Data Flow Diagram - Level 0

DFD level 0 is called a Context Diagram. It is a simple overview of the whole system being modeled. The figure.3 shows the DFD level 0 of the system.

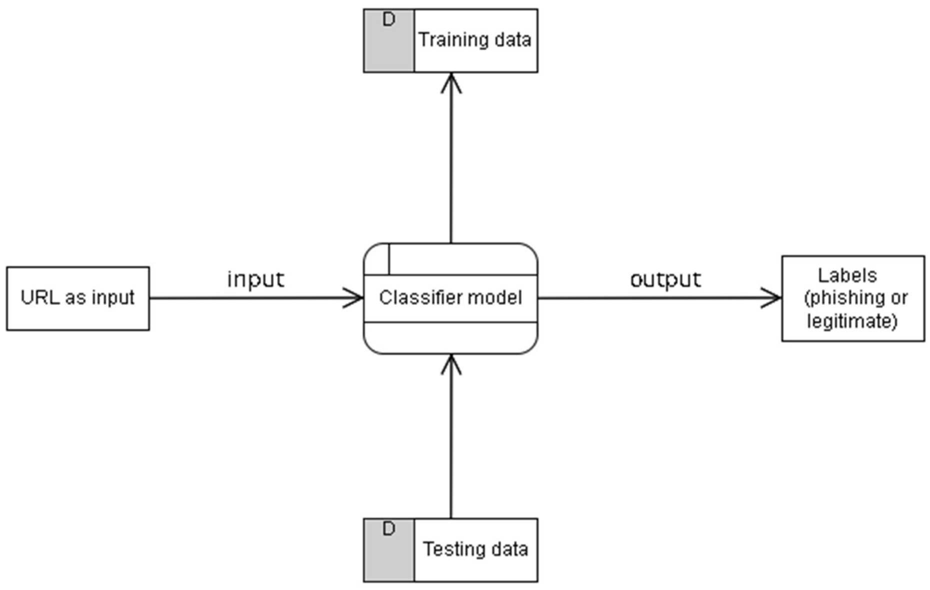


figure 3

It shows the system as a high-level process with its relationship to the external entities. It should be easily acknowledged by a wide range of audience from stakeholders to developers to data analysts.

6.3.2 Data Flow Diagram - Level 1

DFD level 1 gives a more detailed explanation of the Context diagram. The high-level process of the Context diagram is broken down into its subprocesses. The DFD level 1 of the system is depicted in the figure.4

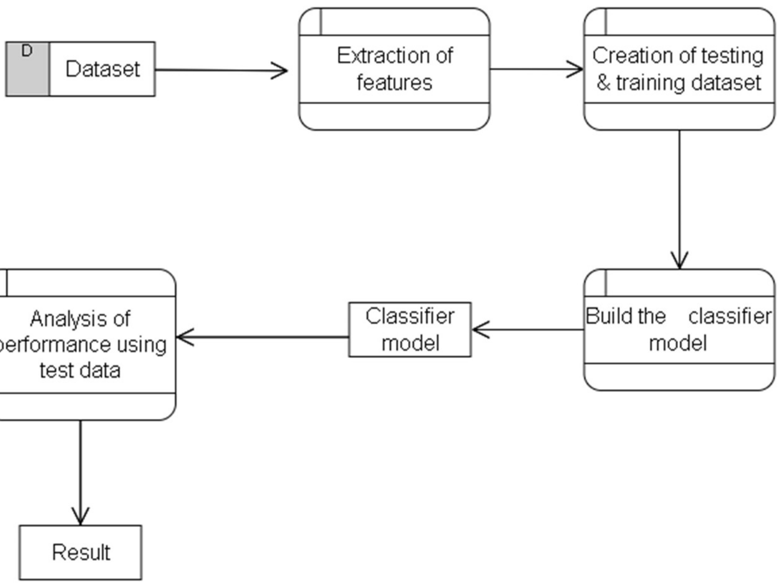

D
Dataset
Extraction of features
Creation of testing & training dataset
Build the classifier model
Classifier model
Analysis of performance using test data
Result

figure.4

The Level 1 DFD takes a step deep by including the processes involved in the system such as feature extraction, splitting of dataset, building the classifier, etc. and hence gives a more detailed vision of the system.

6.3.3 Data Flow Diagram - Level 2

DFD level 2 goes one more step deeper into the subprocesses of Level 1. It might require more text to get into the necessary level of detail about the functioning of the system.

The Level 2 gives a more detailed view of the system by categorizing the processes involved in the system to three categories namely preprocessing, feature scaling and classification. It also graphically depicts each of these categories in detail and gives a complete idea of how the system works.

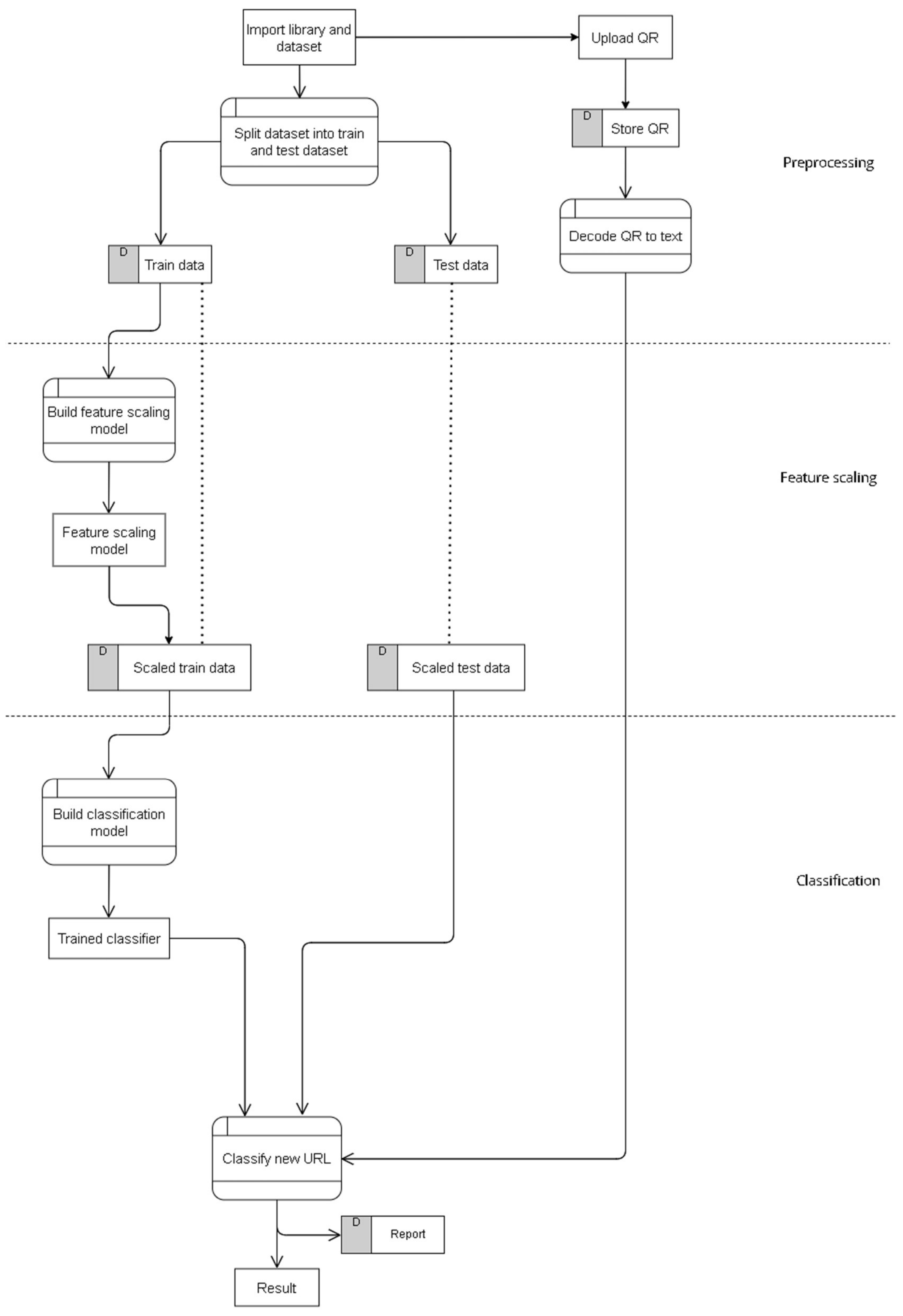

Import library and dataset
Upload QR
D Store QR
Split dataset into train and test dataset
Preprocessing
Decode QR to text
D Train data
D Test data
Build feature scaling model
Feature scaling
Feature scaling model
D Scaled train data
D Scaled test data
Build classification model
Classification
Trained classifier
Classify new URL
D Report
Result

figure.5

6.4 ACTIVITY DIAGRAM

Activity diagram is a behavioral diagram. It depicts the control flow from a start point to an end point showing various paths which exists during the execution of the activity

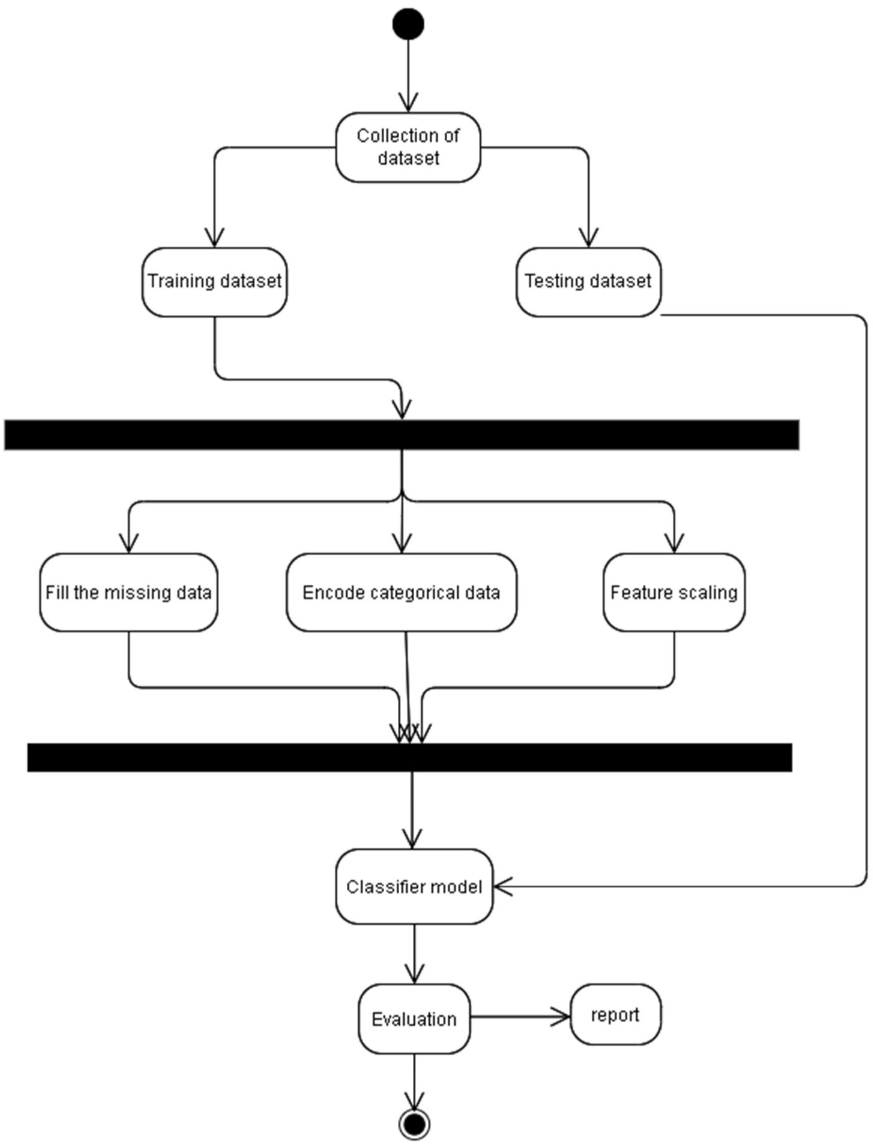
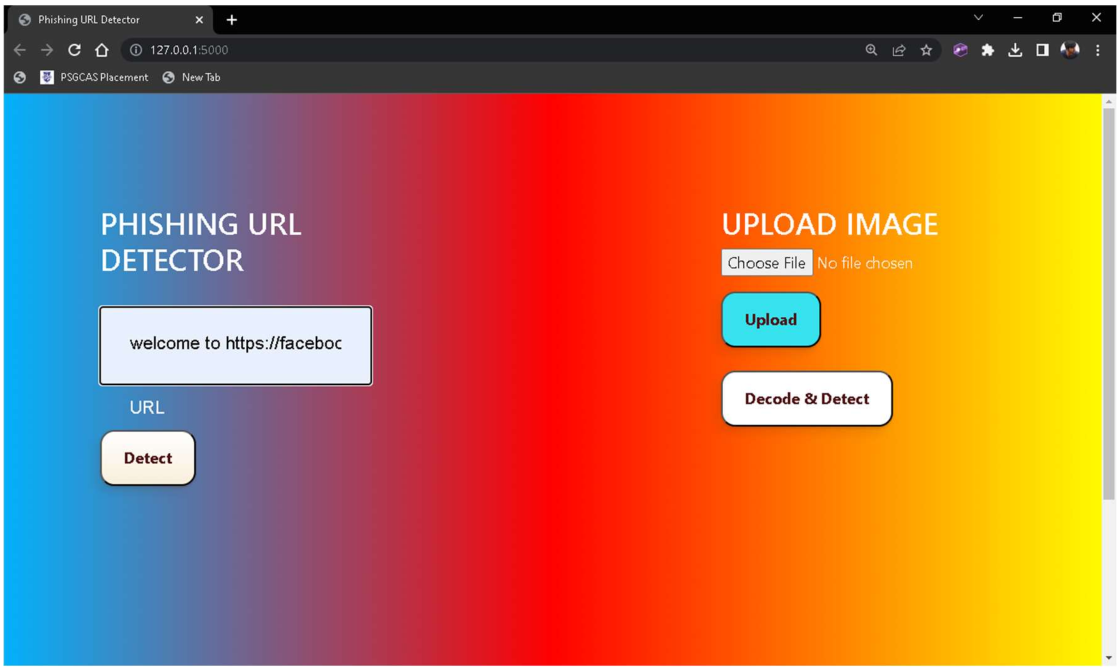


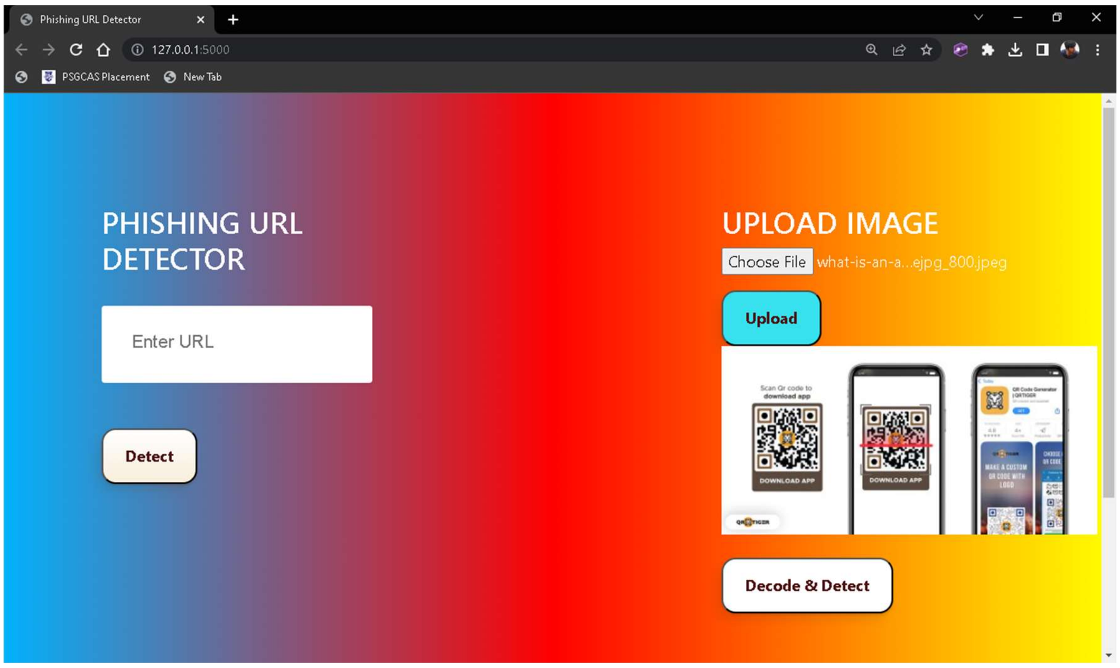
figure.6

6.5 INPUT DESIGN

6.5.1 Input design 1

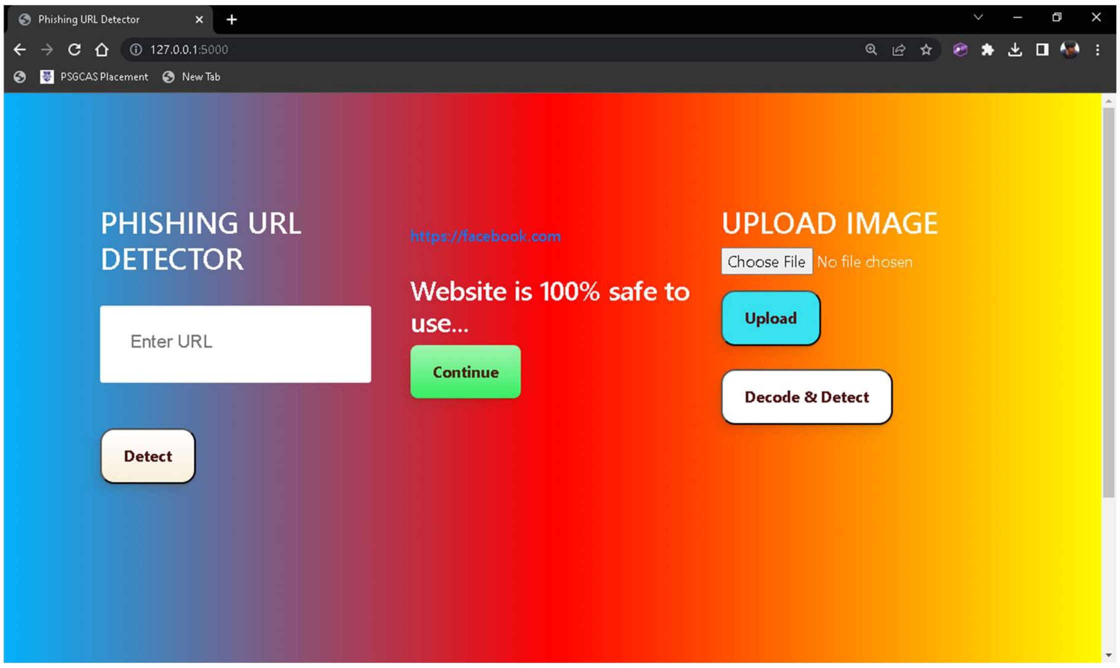


6.5.2 Input design 2

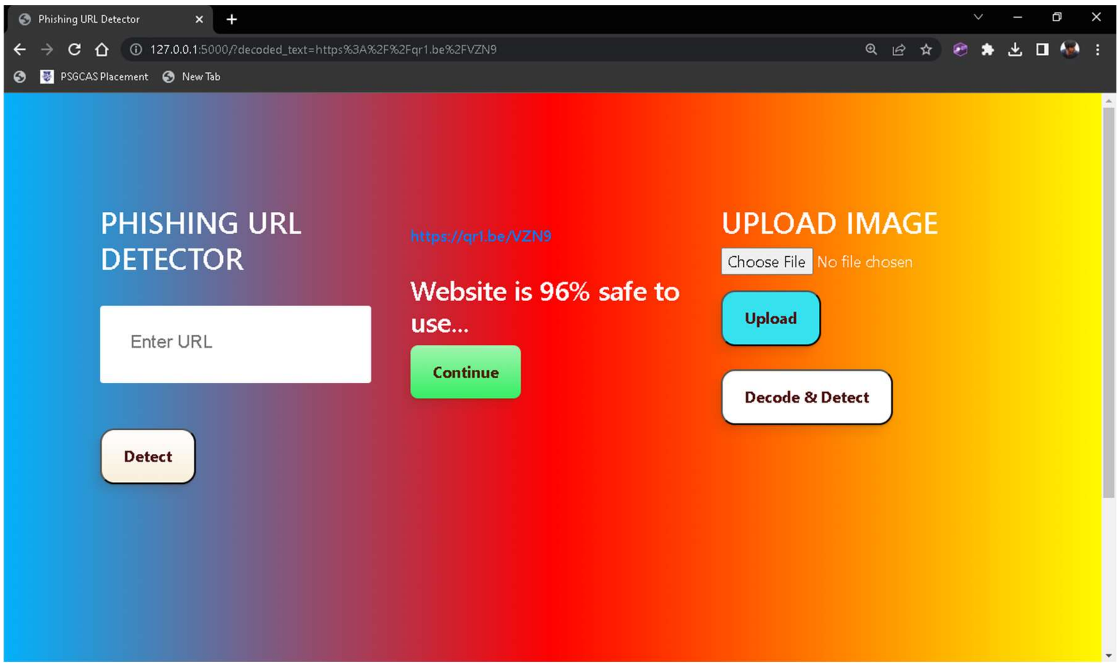


6.6 OUTPUT DESIGN

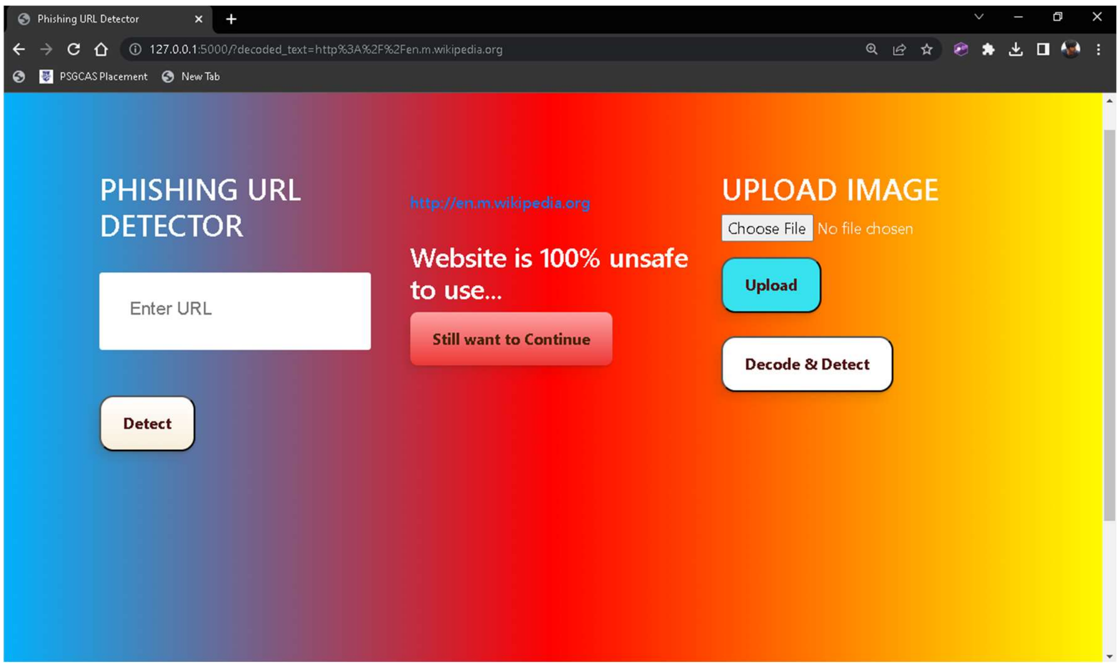
6.6.1 Output design 1



6.6.2 Output design 2



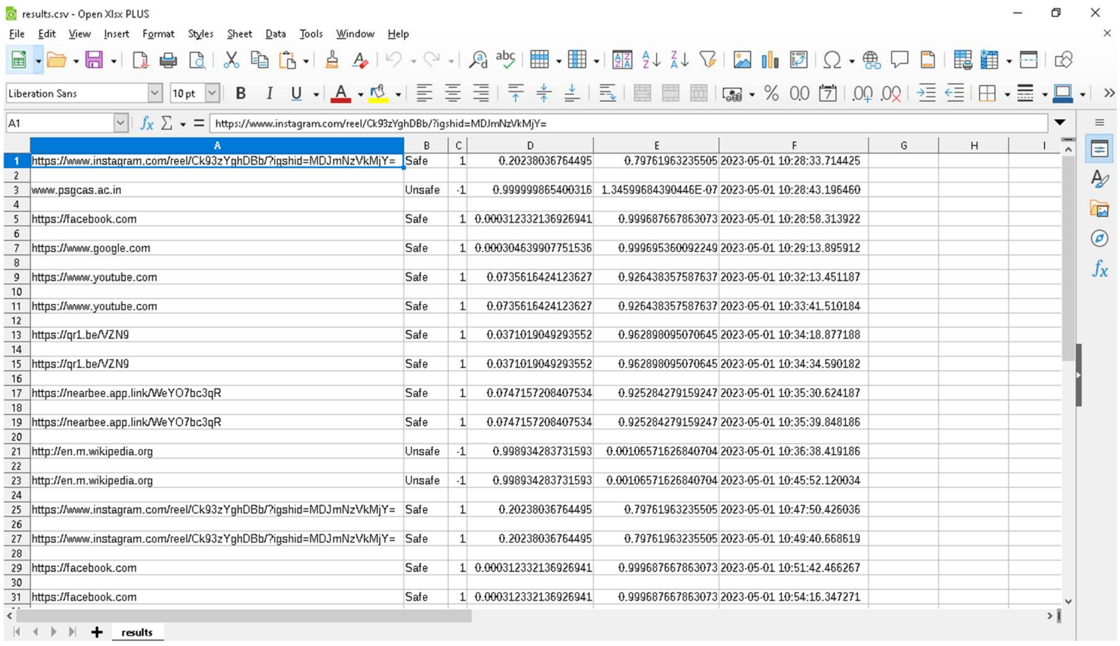
6.6.3 Output design 3



Website is 100% unsafe to use...

6.7 REPORT DESIGN

6.7.1 Report design


results.csv - Open Xlsx PLUS
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
A1
results

https://www.instagram.com/reel/Ck93zYghDBb/?igshid=MDJmNzVkMjY=

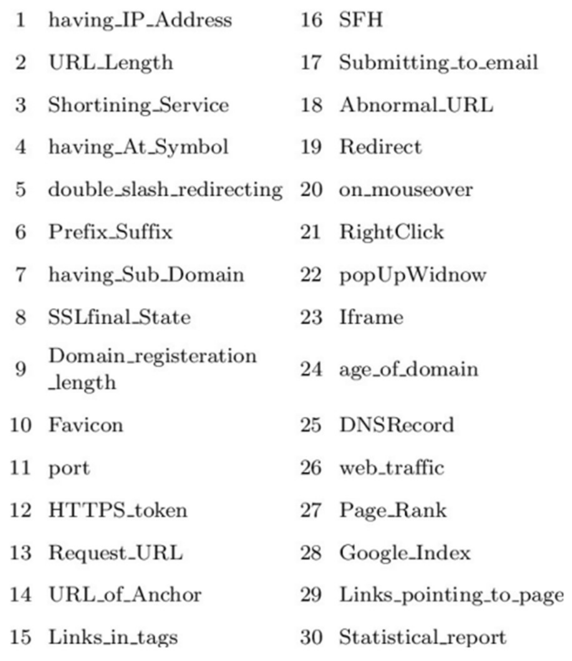
	A	B	C	D	E	F
1	https://www.instagram.com/reel/Ck93zYghDBb/?igshid=MDJmNzVkMjY=	Safe	1	0.20238036764495	0.79761963235505	2023-05-01 10:28:33.714425
2						
3	www.psgcas.ac.in	Unsafe	-1	0.999999865400316	1.34599684390446E-07	2023-05-01 10:28:43.196460
4						
5	https://facebook.com	Safe	1	0.000312332136926941	0.999687667863073	2023-05-01 10:28:58.313922
6						
7	https://www.google.com	Safe	1	0.000304639907751536	0.999695360092249	2023-05-01 10:29:13.895912
8						
9	https://www.youtube.com	Safe	1	0.0735616424123627	0.926438357587637	2023-05-01 10:32:13.451187
10						
11	https://www.youtube.com	Safe	1	0.0735616424123627	0.926438357587637	2023-05-01 10:33:41.510184
12						
13	https://qr1.be/VZN9	Safe	1	0.0371019049293552	0.962898095070645	2023-05-01 10:34:18.877188
14						
15	https://qr1.be/VZN9	Safe	1	0.0371019049293552	0.962898095070645	2023-05-01 10:34:34.590182
16						
17	https://nearbee.app.link/WeYO7bc3qR	Safe	1	0.0747157208407534	0.925284279159247	2023-05-01 10:35:30.624187
18						
19	https://nearbee.app.link/WeYO7bc3qR	Safe	1	0.0747157208407534	0.925284279159247	2023-05-01 10:35:39.848186
20						
21	http://en.m.wikipedia.org	Unsafe	-1	0.998934283731593	0.00106571626840704	2023-05-01 10:36:38.419186
22						
23	http://en.m.wikipedia.org	Unsafe	-1	0.998934283731593	0.00106571626840704	2023-05-01 10:45:52.120034
24						
25	https://www.instagram.com/reel/Ck93zYghDBb/?igshid=MDJmNzVkMjY=	Safe	1	0.20238036764495	0.79761963235505	2023-05-01 10:47:50.426036
26						
27	https://www.instagram.com/reel/Ck93zYghDBb/?igshid=MDJmNzVkMjY=	Safe	1	0.20238036764495	0.79761963235505	2023-05-01 10:49:40.668619
28						
29	https://facebook.com	Safe	1	0.000312332136926941	0.999687667863073	2023-05-01 10:51:42.466267
30						
31	https://facebook.com	Safe	1	0.000312332136926941	0.999687667863073	2023-05-01 10:54:16.347271

7. SYSTEM TESTING AND IMPLEMENTATION

7.1 IMPLEMENTATION PROCESS

The first step of the research work was determining the right data set. The dataset selected was collected from Kaggle for this task. The reasons behind selecting this dataset are several. It includes:

- The data set is large, so working with it is intriguing.
- The number of features in the data set is 30 giving a wide range of features making the predictions a little more accurate. The below figure shows the features being considered.



1	having_IP_Address	16	SFH
2	URL_Length	17	Submitting_to_email
3	Shortining_Service	18	Abnormal_URL
4	having_At_Symbol	19	Redirect
5	double_slash_redirecting	20	on_mouseover
6	Prefix_Suffix	21	RightClick
7	having_Sub_Domain	22	popUpWidnow
8	SSLfinal_State	23	Iframe
9	Domain_registeration _length	24	age_of_domain
10	Favicon	25	DNSRecord
11	port	26	web_traffic
12	HTTPS_token	27	Page_Rank
13	Request_URL	28	Google_Index
14	URL_of_Anchor	29	Links_pointing_to_page
15	Links_in_tags	30	Statistical_report

- The number of URLs is quite evenly distributed among the 2 categories.
- **Splitting** the dataset into training part of dataset and testing part of dataset. The dataset was split into training and testing dataset with 75% for training and 25% for testing using the "train test split" method. The splitting was done after assigning the dependent variables and independent variables.
- **Preprocessing** involves filling the missing data or removing the missing data and getting a clean dataset. But the dataset chosen was already preprocessed and did not require any further preprocessing from my end. The only step to be performed in preprocessing was feature scaling.

- **Feature Scaling** is a procedure to normalize the independent variable present in the information in a fixed range. It is performed during the data pre-processing to deal with varying magnitudes. There are two ways of feature scaling – Normalization and Standardization. The project uses standardization feature scaling methods. The variables should be put in the same scale, else one variable might dominate others hence might affect the result.
- **Standardization** is another scaling procedure where the values are based on the mean with a unit standard deviation. This implies the mean of that attribute gets zero and the resultant distribution has a unit standard deviation.

  $$Xstd = (x - \text{mean}(x))/ \text{ standard deviation}(x)$$
- **Normalization** is a scaling method where values are moved and resized so they wind up going somewhere in the range of 0 and 1. It is otherwise called Min-Max scaling.

  $$Xnorm = (x - \min(x))/(\max(x) - \min(x))$$

  The project uses StandardScaler. It fits and transforms only the independent variables. The dependent variables need not be scaled in the classification method. The dummy variables which we get from categorical data may or may not be scaled depending on context.
- **Feature Extraction:** Feature values are extracted using python modules like whois, requests, socket, re, ipaddress, BeautifulSoup, etc. to get information regarding ip address, length of url, domain name, subdomains, presence of favicon, etc. The value obtained is stored in a list. This is being done because the dataset is in this format and hence the classifier will be trained with input of this format. Therefore, when a URL is passed as input to the system, it converts it into a python list of 30 elements each representing its respective feature and thereafter that list is fed to the trained classifier. The classifier that is being used includes KNN, kernel SVM, Decision Tree and random forest classifier.

7.2 CLASSIFIERS

7.2.1 sklearn.neighbors.KNeighborsClassifier

Classifier implementing k-nearest neighbors.

Parameters used:

- N neighbors : It is the number of neighbors to be considered while categorizing and was considered 5 in the algorithm.
- Metric : It depicts the distance metric to be used. The one used in the algorithm is ‘minkowski’
- P : It is the power parameter for the metric. The algorithm uses p = 2 which is equivalent to Euclidean distance

7.2.2 sklearn.tree.DecisionTreeClassifier

Classifier that is used to implement a decision tree.

Parameters used:

- criterion : the function that is used to measure the quality of a split. The one that is used in the algorithm is “entropy”

7.2.3 sklearn.ensemble.RandomForestClassifier

Classifier that is used to implement random forest classifier. Random forest,as the name implies, constitutes of many separate decision trees which all works as an ensemble Each separate tree of the Random forest gives out a class forecast and the class with the most votes transforms into our model’s desire as

Parameters used:

- N estimators : The number of trees in the forest. The number used in the algorithm is 10.
- criterion : the function that is used to measure the quality of a split. The one that is used in the algorithm is “entropy”.

7.2.4 sklearn.linear_model.LogisticRegression

Logistic regression predicts the output of a categorical variable. Therefore the outcome must be a categorical or discrete value. Logistic Regression is much similar to

Linear Regression except that how they use Linear Regression is used for solving Regression problems, whereas Logistic Regression is used for solving classification problems.

Parameters used:

- log.predict : It predicts the target value from the model for the samples.
- Metric : It depicts the distance metric to be used. The one used in the algorithm is 'minkowski'

7.2.5 sklearn.ensemble.GradientBoostingClassifier

Gradient boosting classifiers are a group of machine learning algorithms that contain many weak learning models together to create a strong predictive model. Decision trees are usually used when gradient boosting algorithms play a crucial role in dealing with the bias variance trade-off. Unlike bagging algorithms, which only control high variance in a model, boosting controls both the aspects (bias & variance), and is considered to be more effective.

Parameters used:

- GradientBoostingClassifier : classifier combines several weak learning models to produce a powerful predicting model.
- Metric : It depicts the distance metric to be used. The one used in the algorithm is 'minkowski'

7.3 TESTING AND VALIDATION

Here, we check for the working of the proposed system by testing and comparing the result of the algorithm and the actual result. It is basically validating the system. The testing is done for each algorithm with a legitimate and phishing URL and the results are as follows.

7.3.1 Unit Testing

Unit Testing is a testing approach where the units of the modules are investigated to check regardless of whether they are fit as a fiddle to be utilized.

7.3.1.1 Unit testing of KNN algorithm - 1

Test case	01
Test name	"Testing of KNN - 1"
Input	http://crikster.co.za/altcustomer CARD/altCustomerB/images/js.php "c ></script><script type="text/javascript"> var siteURL = 'http://crikster.co.za/altcustom
Expected output	Phishing
Actual output	Phishing
Remark	Success

Table4

7.3.1.2 Unit testing of KNN algorithm - 2

Test case	02
Test name	"Testing of KNN - 2"
Input	https://twitter.com/login
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table5

7.3.1.3 Unit testing for Logistic Regression - 1

Test case	03
Test name	"Testing of LoR- 2"
Input	myuniversity.edurenewal.com
Expected output	Phishing
Actual output	Phishing
Remark	Success

Table6

7.3.1.4 Unit testing Logistic Regression - 2

Test case	04
Test name	“Testing of LoR- 2”
Input	https://www.udemy.com/
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table7

7.3.1.5 Unit testing for Decision tree algorithm - 1

Test case	05
Test name	“Testing of Decision tree -1”
Input	paypal.de@secure-server.de/secureenvironment
Expected output	Phishing
Actual output	Phishing
Remark	Success

Table8

7.3.1.6 Unit testing for Decision tree algorithm - 2

Test case	06
Test name	“Testing of Decision tree -2”
Input	https://www.wikipedia.org/
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table9

7.3.1.7 Unit Testing of RFC algorithm -1

Test case	07
Test name	“Testing of RFC -1”
Input	http://63.17.167.23/pc/ verification.htm?=https://www.paypal .com/
Expected output	Phishing
Actual output	Phishing
Remark	Success

Table10

7.3.1.8 Unit testing of RFC algorithm - 2

Test case	08
Test name	“Testing of RFC -2”
Input	https://calendar.google.com/calendar/r
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table11

7.3.1.9 Unit testing of GBC algorithm - 1

Test case	09
Test name	“Testing of GBC -1”
Input	http://cabinetkignima.com/Wellsfargo_keys_account8/page2.html
Expected output	Phishing
Actual output	Phishing
Remark	Success

table12

7.3.1.10 Unit testing of GBC algorithm -2

Test case	10
Test name	“Testing of GBC -2”
Input	https://www.psgcas.ac.in
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table13

7.3.1.11 Unit testing of QR upload & decode

Test case	11
Test name	“QR upload & decode”
Input	Upload image
Expected output	Legitimate
Actual output	Legitimate
Remark	Success

Table14

7.3.2 Integration testing

Integration Testing is a testing approach where the units of the modules are integrated and then investigated to check regardless of whether they are fit to be utilized.

7.3.2.1 Importing modules

Test case	12
Test name	“ Importing modules ”
Input	import ‘module’ statement
Expected output	The module to be imported
Actual output	The module was imported and could be used
Remark	Success

Table15

7.3.2.2 Importing dataset

Test case	13
Test name	“ Importing dataset ”
Input	Import “ dataset ” statement
Expected output	The dataset to be imported
Actual output	The dataset was imported and could be used
Remark	Success

Table16

7.3.2.3 Importing user defined functions

Test case	14
Test name	“ Importing user defined Functions ”
Input	Import “ Extraction ” function
Expected output	The function to be imported that returns a list
Actual output	The function was imported and returned the list as expected
Remark	Success

Table17

7.3.3 System Testing

System testing is a testing approach that checks for completely integrated system's validation.

7.3.3.1 URL

Test case	15
Test name	" System testing - URL "
Input	Sample URL provided to check whether it is a phishing or legitimate URL
Expected output	All the modules like importing of modules, dataset and functions defined and provide the result
Actual output	The application reacts as expected
Remark	Success

Table18

7.3.3.2 QR code

Test case	16
Test name	" System testing - QR "
Input	Sample QR image is provided to check whether it contains links or not. The link is provided to model for prediction.
Expected output	The link and prediction about that link has to display
Actual output	The link and prediction is displayed
Remark	Success

Table19

8. SCOPE OF FUTURE ENHANCEMENT

Further work can be done to enhance the model by using ensembling models to get a greater accuracy score. Ensemble methods is a ML technique that combines many base models to generate an optimal predictive model. Further reaching future work would be combining multiple classifiers, trained on different aspects of the same training set, into a single classifier that may provide a more robust prediction than any of the single classifiers on their own. The project can also include other variants of phishing like smishing, vishing, etc. to complete the system. Looking even further out, the methodology needs to be evaluated on how it might handle collection growth. The collections will ideally grow incrementally over time so there will need to be a way to apply a classifier incrementally to the new data, but also potentially have this classifier receive feedback that might modify it over time.

9. CONCLUSION

The demonstration of phishing is turning into an advanced danger to this quickly developing universe of innovation. Today, every nation is focusing on cashless exchanges, business online, tickets that are paperless and so on to update with the growing world. Yet phishing is turning into an impediment to this advancement. Individuals are not feeling the web is dependable now. It is conceivable to utilize AI to get information and assemble extraordinary information items. A lay person, completely unaware of how to recognize a security danger, shall never invite the danger of making money related exchanges on the web. Phishers are focusing on installment industry and cloud benefits the most.

The project means to investigate this region by indicating an utilization instance of recognizing phishing sites utilizing ML. It aimed to build a phishing detection mechanism using machine learning tools and techniques which is efficient, accurate and cost effective. The project was carried out in a Jupyter notebook and was written in Python. The proposed method used five machine learning classifiers to achieve this and a comparative study of the five algorithms was made. A good accuracy score was also achieved. The five algorithms used are K-Nearest neighbor, Logistic Regression, Decision Tree, Random Forest Classifier, and Gradient Boosting Classifier. All the five classifiers gave promising results with the best being Gradient Boosting Classifier with an accuracy score of 97.4%. The accuracy score might vary while using other datasets and other algorithms might provide better accuracy than Gradient Boosting classifier. Gradient boosting classifier is an ensemble classifier and hence the high accuracy. This model can be deployed in real time to detect the URLs as phishing or legitimate.

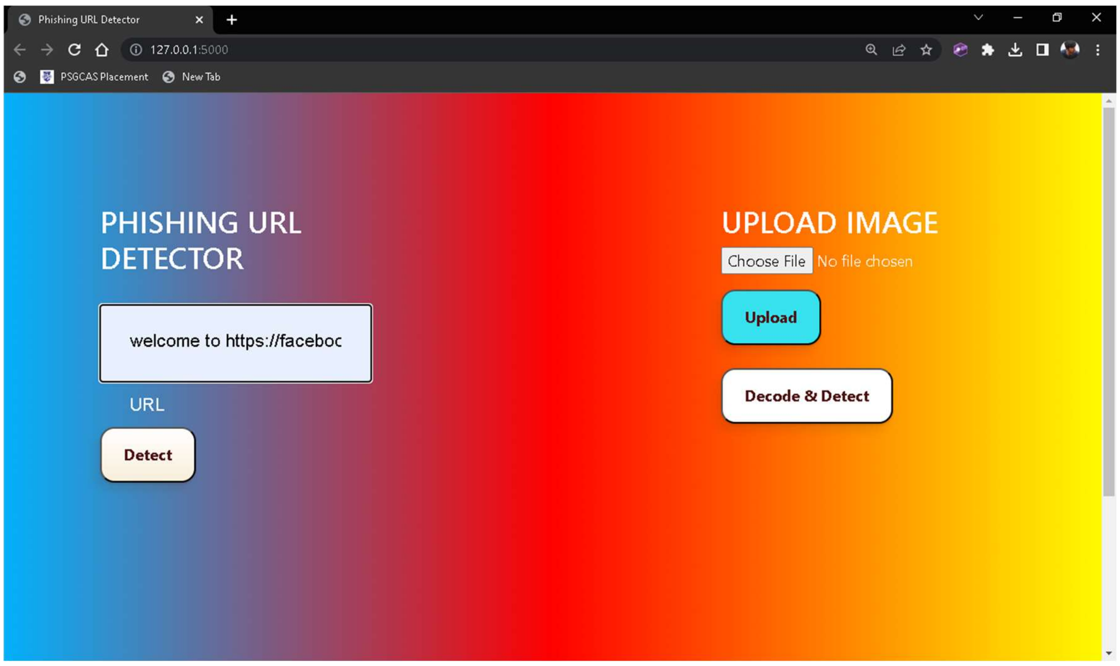
10. BIBLIOGRAPHY

1. https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d
2. https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-bd375d14eec7
3. https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200
4. https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac
5. https://towardsdatascience.com/understanding-random-forest-58381e0602d2
6. https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/
6. https://docs.python.org/3/tutorial/modules.html
7. https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/
8. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

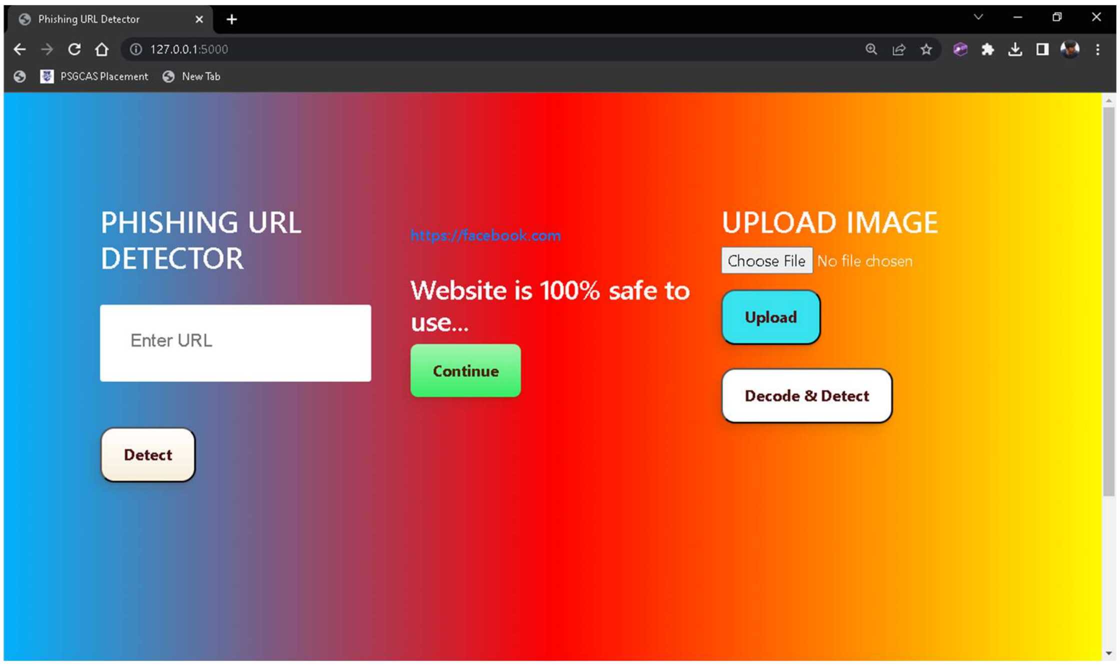
11. APPENDICES

11.1 SCREEN SHOTS

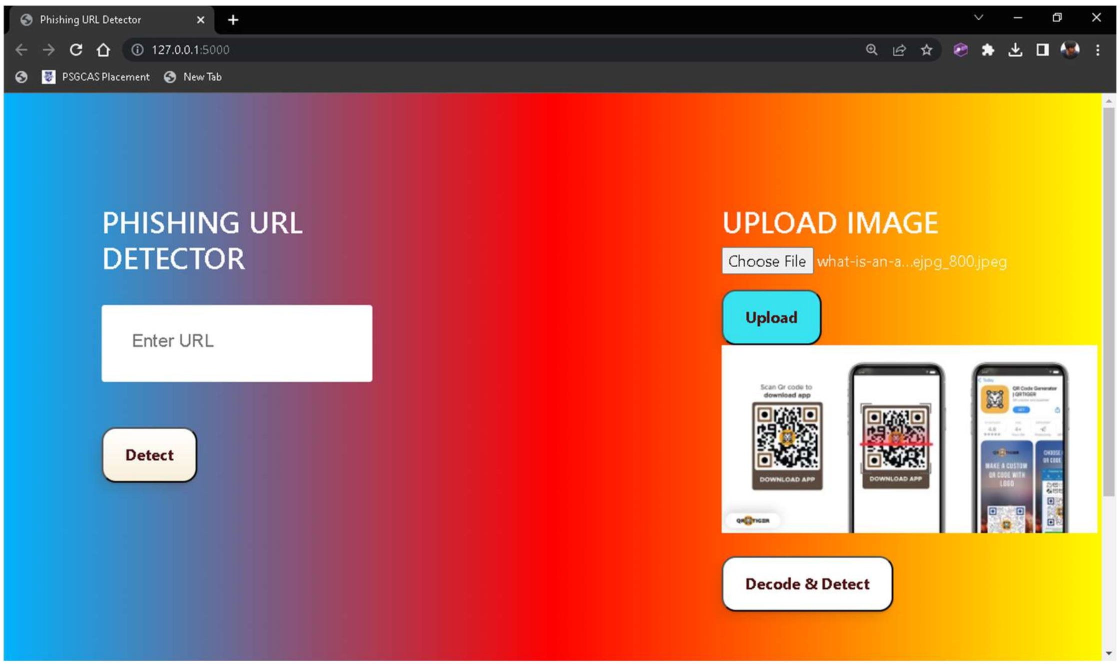
Input



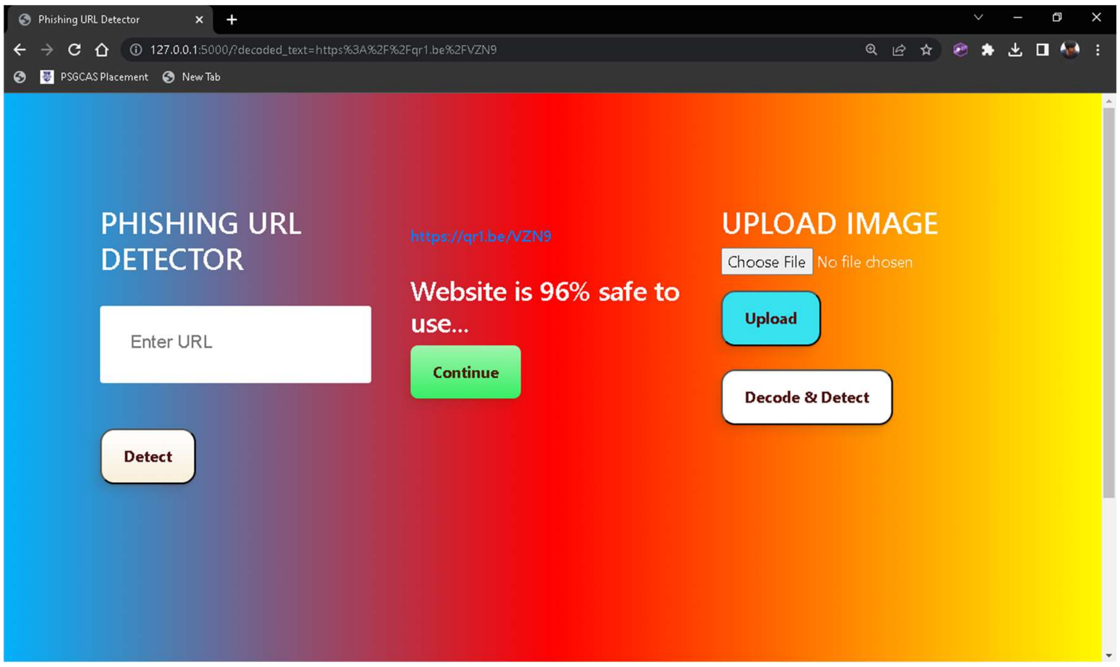
Output



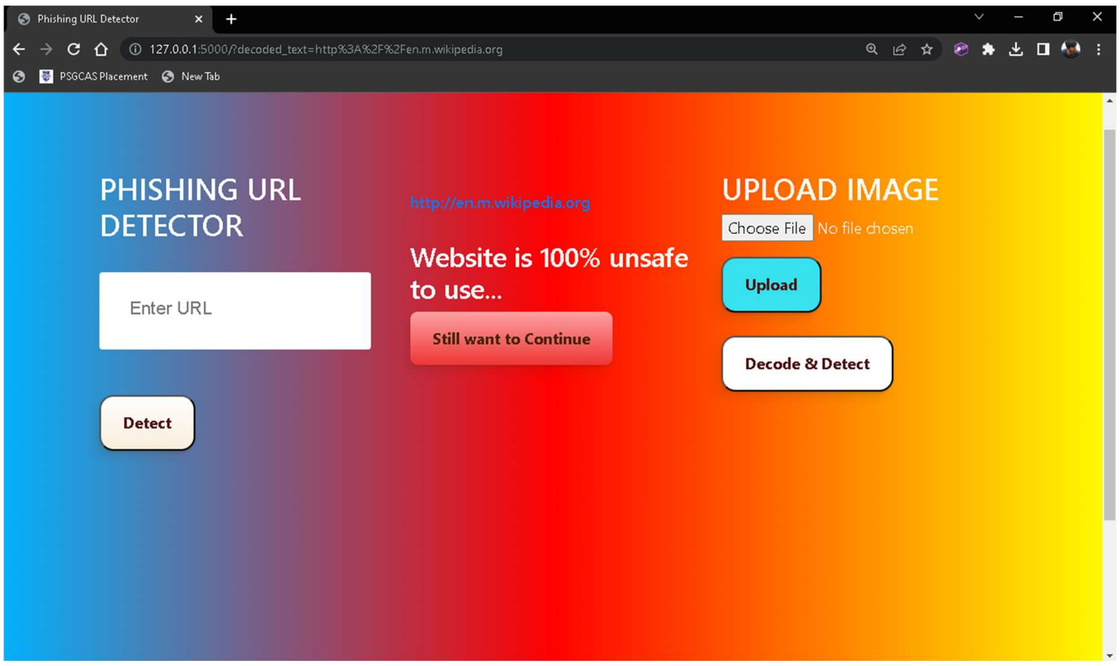
Input



Output

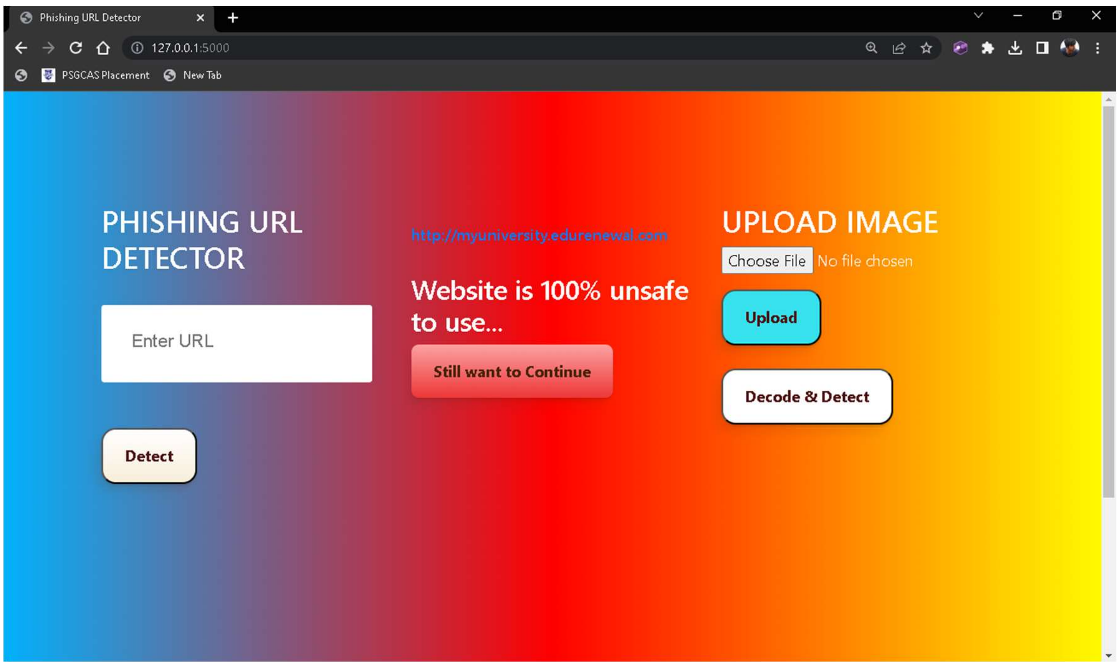


Phishing websites


127.0.0.1:5000/?decoded_text=http%3A%2F%2Fen.m.wikipedia.org

http://en.m.wikipedia.org

Website is 100% unsafe to use...


127.0.0.1:5000

http://myuniversity.edurenewal.com

Website is 100% unsafe to use...

11.2 CODINGS

Application (app.py)

```
#importing required libraries

from flask import Flask, request, render_template, redirect, url_for
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
import os
import re
import datetime
import shutil
import csv
import cv2
from datetime import datetime
import pyzbar.pyzbar as pyzbar
from pyzbar.pyzbar import decode
from werkzeug.utils import secure_filename
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("pickle/model.pkl","rb")
gbc = pickle.load(file)
file.close()


app = Flask(__name__)

@app.route("/", methods=["GET","POST"])
def index():
  if request.method == "POST":
    url = request.form["url"]
    regex=r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-
z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\
]{};:'\".,<>?«»“”‘’]))"
    extracted_url=re.findall(regex,url)
    final_url = ''.join([x[0] for x in extracted_url])
```

```
        if final_url.startswith("http:") or final_url.startswith("www.") or final_url.startswith("https:"):
          obj = FeatureExtraction(final_url)
          x = np.array(obj.getFeaturesList()).reshape(1,30)
          y_pred =gbc.predict(x)[0]
          #1 is safe
          #-1 is unsafe
          y_pro_phishing = gbc.predict_proba(x)[0,0]
          y_pro_non_phishing = gbc.predict_proba(x)[0,1]

         # Save to CSV
        #   save_to_csv(final_url, y_pred, y_pro_phishing, y_pro_non_phishing)
          result = "Safe" if y_pred == 1 else "Unsafe"
          save_to_csv(final_url, result,y_pred,y_pro_phishing,y_pro_non_phishing)
          pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
          return render_template('index.html',xx =round(y_pro_non_phishing,2),url=final_url )
        else:
            return "Invalid URL ¯₩_(ツ)_/¯ "

    else:
        decoded_text = request.args.get('decoded_text')
        regex=r"(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-
z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\
]{};:'\".,<>?«»“”‘’]))"
        decoded_url=re.findall(regex,str(decoded_text))
        final_url = ''.join([x[0] for x in decoded_url])
        if final_url.startswith("http:") or final_url.startswith("www.") or final_url.startswith("https:"):
            obj = FeatureExtraction(final_url)
            x = np.array(obj.getFeaturesList()).reshape(1,30)
            y_pred = gbc.predict(x)[0]
            #1 is safe
            #-1 is unsafe
            y_pro_phishing = gbc.predict_proba(x)[0,0]
            y_pro_non_phishing = gbc.predict_proba(x)[0,1]
            result = "Safe" if y_pred == 1 else "Unsafe"
            save_to_csv(final_url, result,y_pred,y_pro_phishing,y_pro_non_phishing)
            # if(y_pred ==1 ):
            pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
            return render_template('index.html',xx =round(y_pro_non_phishing,2),url=final_url )
        else:
            return render_template("index.html", xx =-1)

@app.route("/decode_qr", methods=["POST"])
```

```
def decode_qr():
    image_path = "static/files/1.png"
    image = cv2.imread(image_path)
    decoded_text = ""
    # Decode QR code in the image
    decoded_objs = pyzbar.decode(image)
    for obj in decoded_objs:
        decoded_text += obj.data.decode("utf-8")
    return redirect(url_for('index',decoded_text=decoded_text))

@app.route('/upload', methods=['POST','GET'])
def upload_file():
    uplfl="images_upl/"
    desfl="static/files/"
    uploaded_file = request.files['file']
    filename = secure_filename(uploaded_file.filename)
    uploaded_file.save(os.path.join(uplfl, filename))
    newname="1.png"
    shutil.copyfile(os.path.join(uplfl, filename), os.path.join(desfl, newname))

    return "Image uploaded"


def save_to_csv(url, result,y_pred,y_pro_phishing,y_pro_non_phishing):
    with open('results.csv', mode='a') as csv_file:
        fieldnames = ['url', 'result', 'Prediction','Phishing probability','Non-Phishing probability','date']
        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

        writer.writerow({'url': url, 'result': result,'Prediction':y_pred,'Phishing
probability':y_pro_phishing,'Non-Phishing probability':y_pro_non_phishing,'date': datetime.now()})


if __name__ == "__main__":
    app.run(debug=True)
```

HTML code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="description" content="This website is develop for identify the safety of url.">
   <meta name="keywords" content="phishing url,phishing,cyber security,machine
learning,classifier,python">
   <meta name="author" content="Mr.Monkey">

   <!-- BootStrap -->
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">

   <link href="static/styles.css" rel="stylesheet">
   <title>Phishing URL Detector</title>

</head>

<body>

<div class=" container">
   <div class="row">
      <div class="form col-md" id="form1">
         <h2>PHISHING URL DETECTOR</h2>

         <br>
         <form action="/" method ="post">
            <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL"
required="" />
            <label for="url" class="form__label">URL</label>
            <button class="button" role="button" >Detect </button>
         </form>

   </div>
   <div class="col-md" id="form2">

       <br>
       <h6 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h6>

       <br>
       <h3 id="prediction"></h3>
       <button   class="button2"   id="button2"   role="button"   onclick="window.open('{{url}}')"
target="_blank" >Still want to Continue</button>
       <button   class="button1"   id="button1"   role="button"    onclick="window.open('{{url}}')"
target="_blank">Continue</button>
   </div>
<div class="form col-md" id="form3">
            <form action="/upload" method="post" enctype="multipart/form-data">
                <h2>UPLOAD IMAGE</h2>
                <div class="form-group">
                    <input    class="uploadfile"    type="file"    name="file"    accept="Image/*"
onchange="preview_image(event)">
                </div>
                <button class="button4" id="button4" onclick="uploadImage()">Upload</button>
            </form>
                     <img id="preview" src="" style="display:none">
            <br>
            <form action="/decode_qr" method="post" >
              <button class="button5" id="button5" onclick="decodeQR()">Decode & Detect</button>
          </form>
        </div>
         <div class="previewimg">
       <body>
         <img src="\static\files\1.png" id="qr_image" style="display: none;">
       </body>
      </div>

</div>
<br>
</div>

   <!-- JavaScript -->
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
       integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
```

```
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>
<script>
function decodeQR() {
  const xhr = new XMLHttpRequest();
  xhr.open("POST", "/decode_qr");
  xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
  xhr.onload = function() {
    const decodedText = JSON.parse(xhr.responseText).decoded_text;
    const xhr2 = new XMLHttpRequest();
    xhr2.open("POST", "/");
    xhr2.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    xhr2.onload = function() {
      const result = JSON.parse(xhr2.responseText).result;
      document.getElementById("prediction").textContent = result;
      document.getElementById("form2").style.display = "block";
    };
    xhr2.send(JSON.stringify({url: decodedText}));
  };
  xhr.send();
}
</script>

<!-- Preview Image -->
<script>
function preview_image(event) {
  var reader = new FileReader();
  reader.onload = function() {
    var preview = document.getElementById('preview');
    var container = document.getElementById('preview-container');
    var img = new Image();
    img.src = reader.result;
    img.onload = function() {
      var canvas = document.createElement('canvas');
      var ctx = canvas.getContext('2d');
      var MAX_WIDTH = 400;
      var MAX_HEIGHT = 300;
      var width = img.width;
      var height = img.height;
      if (width > height) {
        if (width > MAX_WIDTH) {
          height *= MAX_WIDTH / width;
          width = MAX_WIDTH;
        }
      } else {
        if (height > MAX_HEIGHT) {
          width *= MAX_HEIGHT / height;
          height = MAX_HEIGHT;
        }
      }
      canvas.width = width;
      canvas.height = height;
      ctx.drawImage(img, 0, 0, width, height);
      preview.src = canvas.toDataURL('image/jpeg');
      preview.style.display = 'block';
      container.style.width = width + 'px';
      container.style.height = height + 'px';
    }
  }
  reader.readAsDataURL(event.target.files[0]);
}
</script>


   <script>

        let x = '{{xx}}';
        let num = x*100;
        if (0<=x && x<0.50){
            num = 100-num;
        }
        let txtx = num.toString();
        if(x<=1 && x>=0.50){
            var label = "Website is "+txtx +"% safe to use...";
            document.getElementById("prediction").innerHTML = label;
            document.getElementById("button1").style.display="block";
        }
      else if (0<=x && x<0.50){
        var label = "Website is "+txtx +"% unsafe to use..."
        document.getElementById("prediction").innerHTML = label ;
        document.getElementById("button2").style.display="block";
      }

  </script>
<script>
function uploadImage() {
  fileInput = document.getElementById('fileInput');
  file = fileInput.files[0];
  formData = new FormData();
  formData.append('file', file);

  fetch('/upload', {
    method: 'POST',
    body: formData
  }).then(response => {
    // Handle response from server
  });
 }
</script>

</body>

</html>
```