AI / ML SYSTEM TO DETECT PHISHING DOMAINS (URL)

A PROJECT REPORT
Submitted by

RAHUL KUMAR	**(730920201035)**
RAJ LAXMI	**(730920201036)**
RAJNANDAN KUMAR	**(730920201038)**
VICKY KUMAR SINGH	**(730920201052)**

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

EXCEL ENGINEERING COLLEGE
(AUTONOMOUS)

Komarapalayam - 637303

APRIL 2024

EXCEL ENGINEERING COLLEGE(AUTONOMOUS), KOMARAPALAYAM

BONAFIDE CERTIFICATE

Certified that this Project report "**AI / ML SYSTEM TO DETECT PHISHING DOMAINS (URL)**" is the bonafide work of **"RAHUL KUMAR (730920201035), RAJ LAXMI (730920201036), RAJNANDAN KUMAR (730920201038), VICKY KUMAR SINGH (730920201052)"** who carried out the Project work under my supervision.

SIGNATURE
Prof. P. JAYAPRABHA, M.E.,
HEAD OF THE DEPARTAMENT,
Department of Artificial Intelligence and Data Science,
Excel Engineering College,
Komarapalayam, 637303.

SIGNATURE
Mrs. S.L. SWARNA, M.TECH.,
ASSISTANT PROFESSOR,
Department of Artificial Intelligence and Data Science,
Excel Engineering College,
Komarapalayam, 637303.

The Project report submitted for the viva-voce held on …………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude in-depth to our institution “**EXCEL ENGINEERING COLLEGE**” for providing us the opportunity to do the project.

We are greatly indebted to our Honorable Chairman **Dr. A.K. NATESAN, M.Com., MBA (NIT)., M.Phil., Ph.D., FTA,** for providing all the necessary facilities for successful completion for the project.

We Express our heartiest gratitude and thanks to our respected Vice Chairman **Dr. N. MATHAN KARTHICK, M.B.B.S., MISTE., PHIE.,** of **EXCEL GROUP OF INSTITUTIONS** for providing all the facilities for successfully completing the project.

We like to place on record to our beloved Principal and Executive Director **Dr. K. BOMMANNA RAJA, M.E., Ph.D.,** for his valuable suggestion in our entire endeavour.

We thank the Head of the Department **Prof. P. JAYAPRABHA, M.E.**, of Artificial Intelligence and Data Science for her guidance, constant inspiration and encouragement.

We wish to express our Heartfelt Thanks and sincere acknowledgment to our respected **Mrs. S.L. SWARNA, M.TECH,** for her encouragement and dedicated guidance.

We take the privilege to record our everlasting and loving thanks to our parents for their kind help and support which render in bringing our project full manner.

ABSTRACT

The AI / ML System To Detect Phishing Domain (URL) is a webpage with user friendly interface which prevent victim to losing their confidential data such as credit/debit card number, internet banking user id and password. An attacker sends out fake messages that look to come from a trusted source which are phishing URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Logistic Regression, Multilayer perceptron, XG boost classifier. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% Logistic Regression and XG boost classifiers respectively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The “AI / ML System To Detect Phishing Domains (URL)” aim to detect the domains which are type of phishing domains with goal of preventing user to loss their personal data or information and provide an interface to user for help each other.

In this project, we provide a user-friendly interface with web Development and Machine Learning technology. The domain is detected with machine learning algorithms.

To combat this menace effectively, the integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies has emerged as a formidable approach.

By leveraging the power of AI/ML, we can develop sophisticated systems capable of detecting and preventing phishing attempts in real-time, thereby safeguarding users against potential cyber threats.

The primary goal of this endeavor is create an AI/ML-driven system that can autonomously identify phishing domains with a high degree of accuracy, while user Enter URLs.

By analyzing vast amounts of data and learning from patterns indicative of phishing behavior, our system aims to stay ahead of evolving cyber threats and provide proactive defense mechanisms.

Key Points of the AI / ML System To Detect Phishing Domains:-

1. **Data Collection and Preprocessing**:
   The foundation of any AI/ML system lies in its data. We will gather a diverse dataset comprising known phishing domains, legitimate websites, and associated features such as URL structure, domain age, SSL certificate status, etc. Preprocessing techniques will be employed to clean and prepare the data for analysis.
2. **Feature Engineering**:
   Effective feature selection and engineering are crucial for enhancing the system's predictive capabilities. We will identify relevant features that characterize phishing domains and extract meaningful information to feed into the ML models. This may include features like domain reputation, lexical analysis of URLs, presence of suspicious keywords, etc.
3. **Machine Learning Models**:
   Various ML algorithms, such as XG boost classifier, supervised learning classifiers (e.g., Random Forest, Support Vector Machines, XG boost classifier), will be trained on the prepared dataset. These models will learn to distinguish between phishing and legitimate domains based on the extracted features. Additionally, unsupervised learning techniques like clustering may be employed to identify anomalous patterns indicative of phishing behavior.
4. **Real-time Detection and Scoring**:
   Once trained, the ML models will be integrated into a real-time detection pipeline. Incoming URLs will be evaluated using the trained models, assigning a phishing likelihood score to each domain. Based on this score, appropriate actions can be taken, such as warning the user, blocking access, or further investigation.

5. **Continuous Learning and Adaptation:**
   The threat landscape is dynamic, with new phishing techniques constantly emerging. Therefore, our system will incorporate mechanisms for continuous learning and adaptation. This involves regularly updating the dataset, retraining the models with new data and fine-tuning algorithms to improve performance over time.

1.1 Scope of project:

The scope for AI/ML systems to detect phishing domains is vast and continuously expanding, driven by the evolving nature of cyber threats and the increasing reliance on digital platforms.

Here are some key areas where AI/ML systems can make a significant impact in detecting phishing domains:-

Scalability, Real-Time Detection, Adaptability, Multimodal-Analysis, User-Centric Approaches, Collaborative Defense Mechanisms..,ect.

1.2 Objective of the project:

To develop an AI / ML intelligent system which detects phishing domain (websites) and to prevent the user to be victim of phishing attack, and lost their confidential data and provide interface to user for help each other by uploading phishing URLs in public dataset.

CHAPTER 2
LITERATURE SURVEY

2.1 Phish-Net: Predictive Blacklisting to Detect Phishing Attacks

This paper proposes Phish-Net, a machine learning-based system that leverages features extracted from domain registration information and website content to detect phishing domains. The authors demonstrate the effectiveness of their approach through empirical evaluation on real-world datasets**.**

2.2 Detecting Phishing Websites Using Machine Learning

This study presents a comparative analysis of various machine learning algorithms for phishing website detection. The authors explore features such as domain age, lexical attributes, and HTML content, and evaluate the performance of algorithms like Random Forest, Support Vector Machines, and Neural Networks.

2.3 A Survey on Machine Learning Techniques for Phishing Detection and Prediction

This survey provides an overview of machine learning techniques employed in phishing detection and prediction systems. It categorizes existing approaches based on feature extraction methods, classification algorithms, and evaluation metrics, offering insights into the current state-of-the-art methodologies.

2.4 Deep Learning-Based Phishing Detection for Web and Mobile Platforms

This research introduces a deep learning-based approach for detecting phishing attacks targeting both web and mobile platforms. The authors propose a hybrid model combining convolutional and recurrent neural networks to analyze website content and user interaction data, achieving high detection accuracy.

2.5 An Ensemble Machine Learning Approach for Phishing Detection

This work proposes an ensemble learning framework for phishing detection, combining multiple base classifiers to enhance prediction accuracy. The authors investigate feature selection methods and ensemble strategies, demonstrating the effectiveness of their approach on benchmark datasets.

2.6 Adaptive Phishing Detection Using Online Learning

This research introduces an adaptive phishing detection system based on online learning algorithms. The system continuously updates its models using streaming data, adapting to emerging phishing techniques in real-time. Experimental results demonstrate the system's efficacy in detecting previously unseen phishing attacks.

2.7 Explainable Artificial Intelligence for Phishing Detection

This paper addresses the interpretability of machine learning models in phishing detection systems. The authors propose an explainable AI framework that provides insights into model predictions, enabling users to understand the rationale behind phishing domain classifications and fostering trust in the system.

2.8 Federated Learning for Collaborative Phishing Detection

This work explores federated learning techniques for collaborative phishing detection across multiple organizations while preserving data privacy. By aggregating model updates from distributed sources, the proposed approach enables effective phishing detection without sharing sensitive information, facilitating collaborative cyber security efforts.

2.9 Phishing Attack Activity

A phishing attack is one of the most serious threats for any organization and in this section, we present the work done on phishing attacks in more depth along with its different types. Initially, were performed on telephone networks also known as Phone Phreaking which is the reason the term “fishing”.

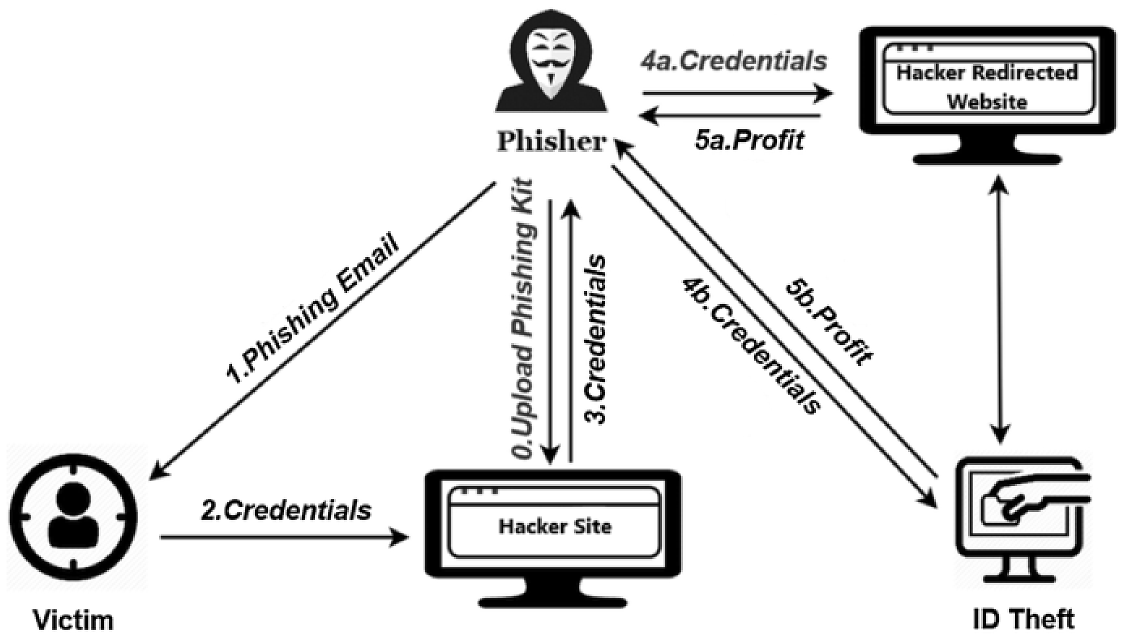


Fig. 2.11 Phishing attack diagram

2.12 Machine learning (ML) for phishing attack detection

ML approaches are popular for phishing websites detection and it becomes a simple classification problem. To train a machine learning model for a learning-based detection system, the data at hand must-have features that are related to phishing and legitimate website classes. Different classifiers are used to detect a phishing attack. Previous studies show that detection accuracy is high as robust ML techniques are used. Several feature selection techniques are used to reduce features. Below figure shows the working of the machine learning model. A batch of input data is given as input for training to the machine learning model to predict the phishing attack or legitimate traffic.

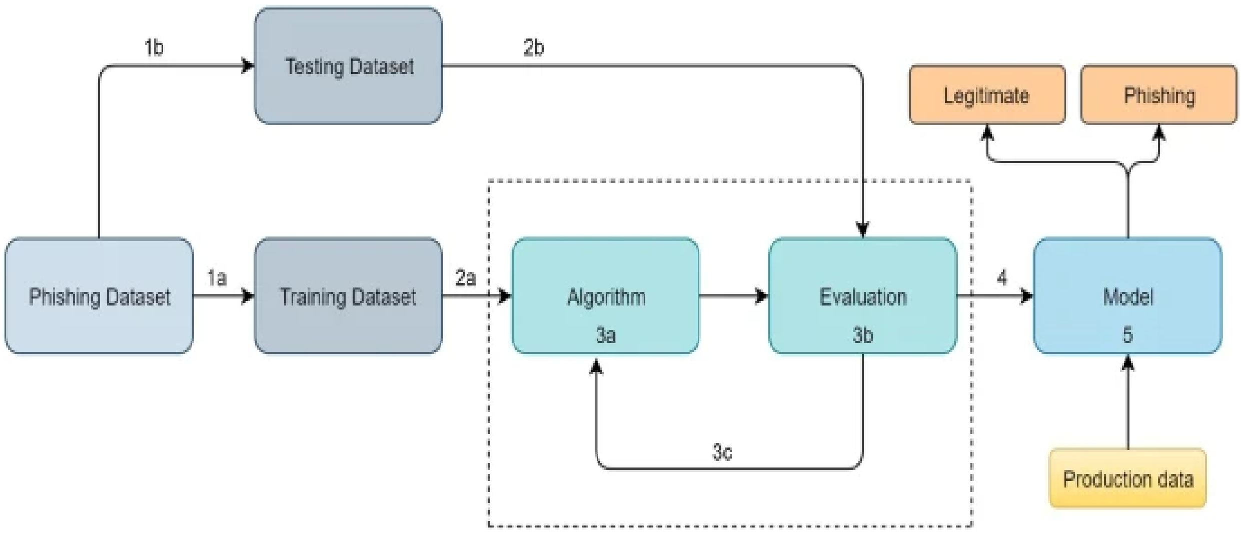


Fig. 2.12 Machine learning for phishing attack detection

CHAPTER 3
EXISTING SYSTEM

There are various system are available for AI/ML systems have been developed to detect phishing domains. Here are some notable examples:

3.1 Confense

- Formery known as phishme, offense a suite of phishing detection phishing detection and response solutions.
- Their platform utilizes machine larning algorithms to analyze email content and identify potential phishing attempts.

3.2 OpenPhish

- Openphish is an open-source project that maintains a database of known phishing websites and URLs.
- It provides APIs and FEEDs that organizations can integrate into their security system to block access to malicious domains.

3.3PhishTank

- PhishTank is a collaborative community that collects and shares information about phishing attacks, including phishing domains.
- It allows users to submit suspected phishing sites and verify reported URLs to help identify and mitigate phishing threats.

3.4 Proofpoint Email Protection

- Proofpoint offers email security solutions that use AI and ML to analyze email behavior identify phishing attempts.
- It includes features for detecting malicious domains and preventing users from interacting with them.

3.5 Detecting Fake Websites

- When users visit a phishing web page that looks like a legitimate website, many people do not remember the legitimate website's domain name, particularly for some start-ups or unknown companies, so users cannot recognize the phishing website based on the URL.
- Some web browsers integrate a security component to detect phishing or malware sites, such as Chrome, which will display warning messages when one visits an unsafe web page.

CHAPTER 4
PROPOSED WORK

4.1 Introduction

This paper presents a comprehensive system designed to detect phishing domains, providing a proactive defense mechanism against these malicious activities. By leveraging advanced technologies such as machine learning, data analytics, and real-time monitoring, our proposed system aims to identify and neutralize phishing threats before they can inflict harm.

Our proposed system begins by collecting a diverse dataset of known phishing domains, sourced from reputable security agencies, public repositories, and proprietary crawlers. Through rigorous feature extraction, including URL characteristics, domain age, SSL certificates, and content analysis, we aim to capture the distinctive traits of phishing domains that differentiate them from legitimate websites.

Integration with web browsers and security plugins facilitates real-time scanning of URLs, empowering users with instant alerts when they encounter potentially malicious websites. Furthermore, our system incorporates mechanisms for community reporting and feedback, harnessing the collective intelligence of users to enhance detection capabilities continually.

Phishing attacks continue to pose a significant threat to individuals and organizations worldwide. The proposed AI/ML phishing domains detection system aims to mitigate this threat by leveraging machine learning techniques to automatically identify and block malicious domains before they can cause harm.

4.2 System Overview

The system will consist of following componenets:-

4.2.1 Data Collection Module

Collects a diverse dataset known phishing domains, legitimate domains and associated metadata.

4.2.2 Feature Extraction Module

Extracts relevant features from domain name, URLs, SSL certificates and other metadata.

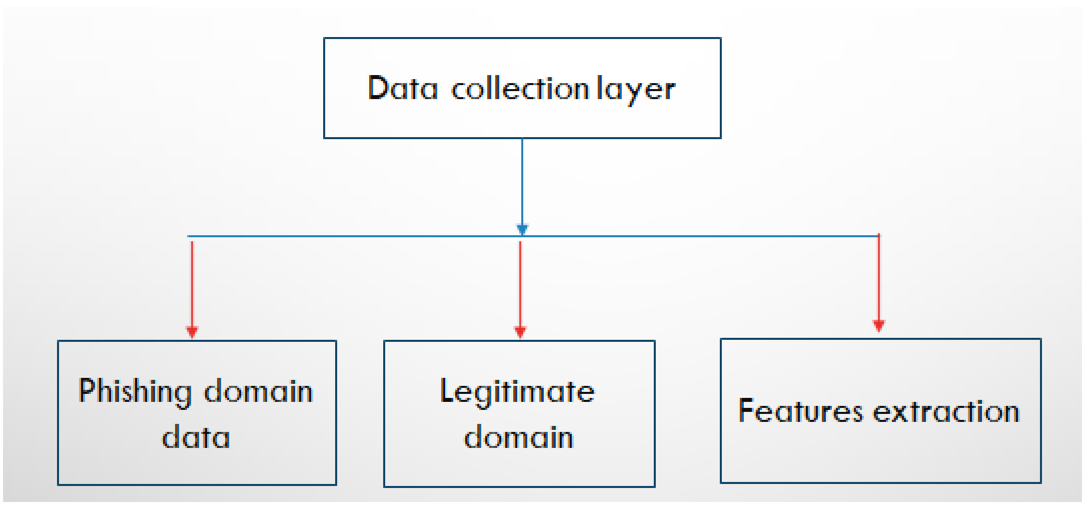
4.2.3 Machine Learning Model

Trains a machine learning model using the extracted features to classify domains as either phishing or legitimate.

4.2.4 Real-Time Detection Engine

Implements the trained model to classify domains in real-time and flag suspicious URLs.

4.3 System Architecture Design



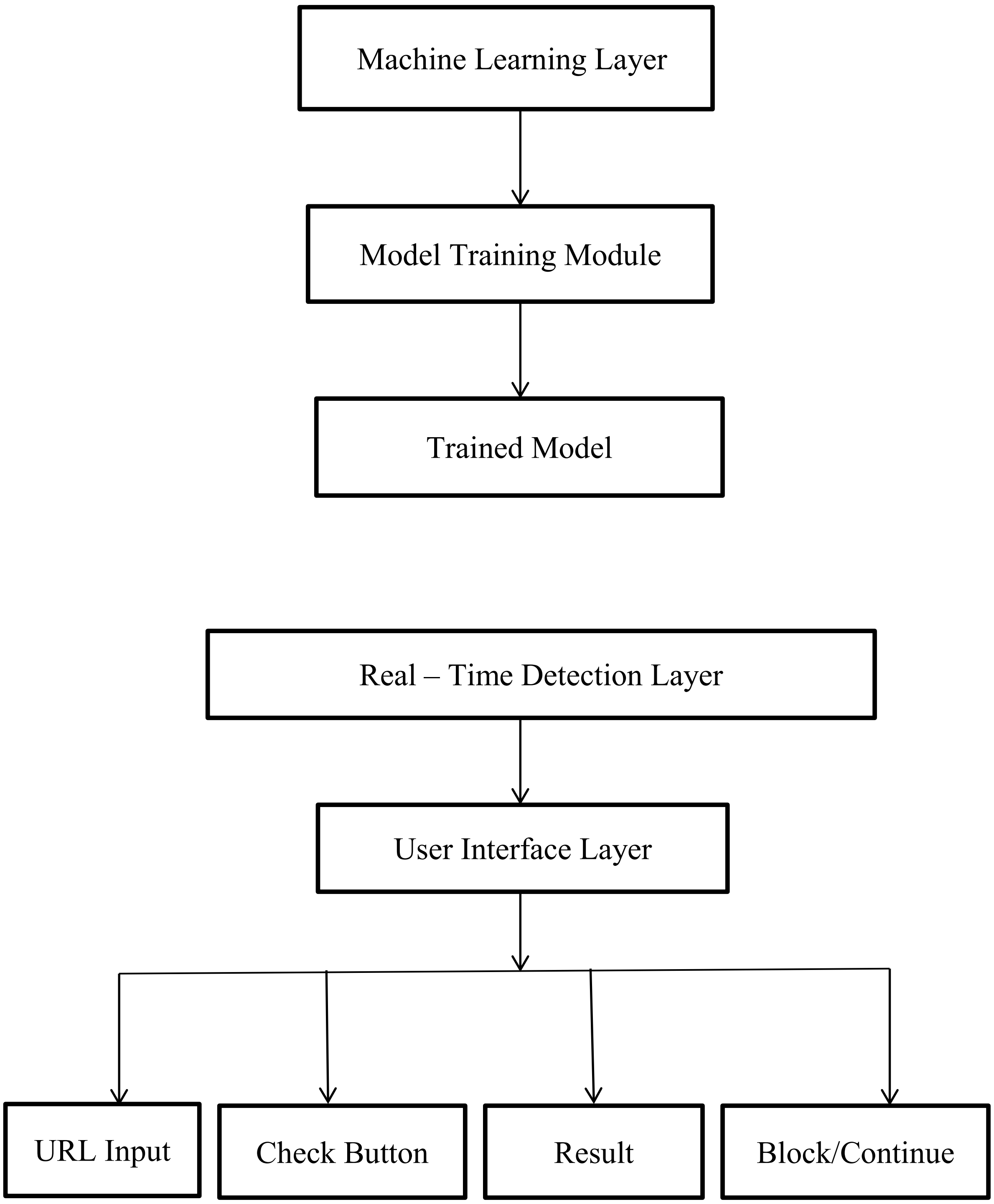


Fig. 4.3 System Architecture Design

4.4 Data Collection

- Gather a large dataset of labeled domain names, including both legitimate and phishing domains. Publicly available datasets like PhishTank can be used, along with additional sources for legitimate domain data.

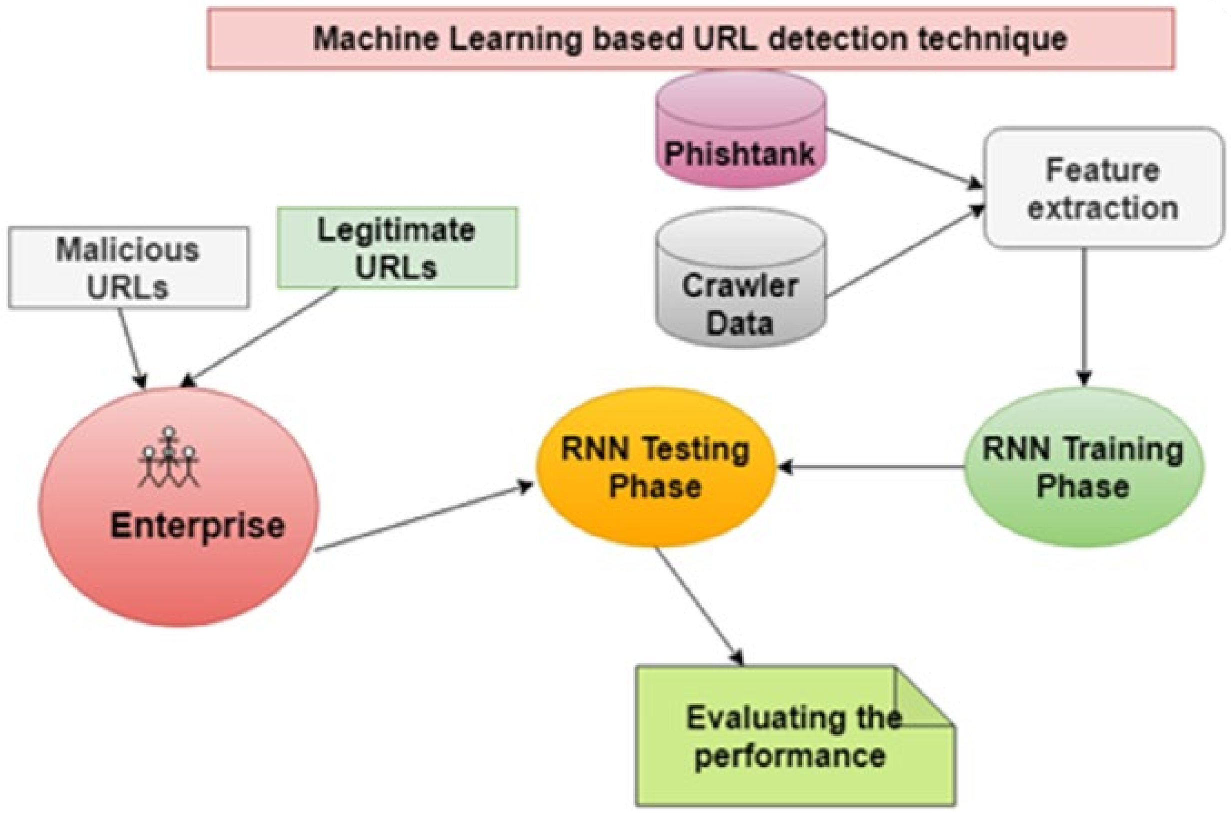


Fig. 4.4 Data for ML URL Detection Technique

4.5 Feature Extraction:

- Extract features from domain names that can be indicative of phishing. Features could include:
    - Domain length
    - Presence of digits or special characters
    - Similarity to popular domains (e.g., "g00gle.com" instead of "google.com")
    - Use of subdomains

- Domain age
- WHOIS information
- SSL certificate information

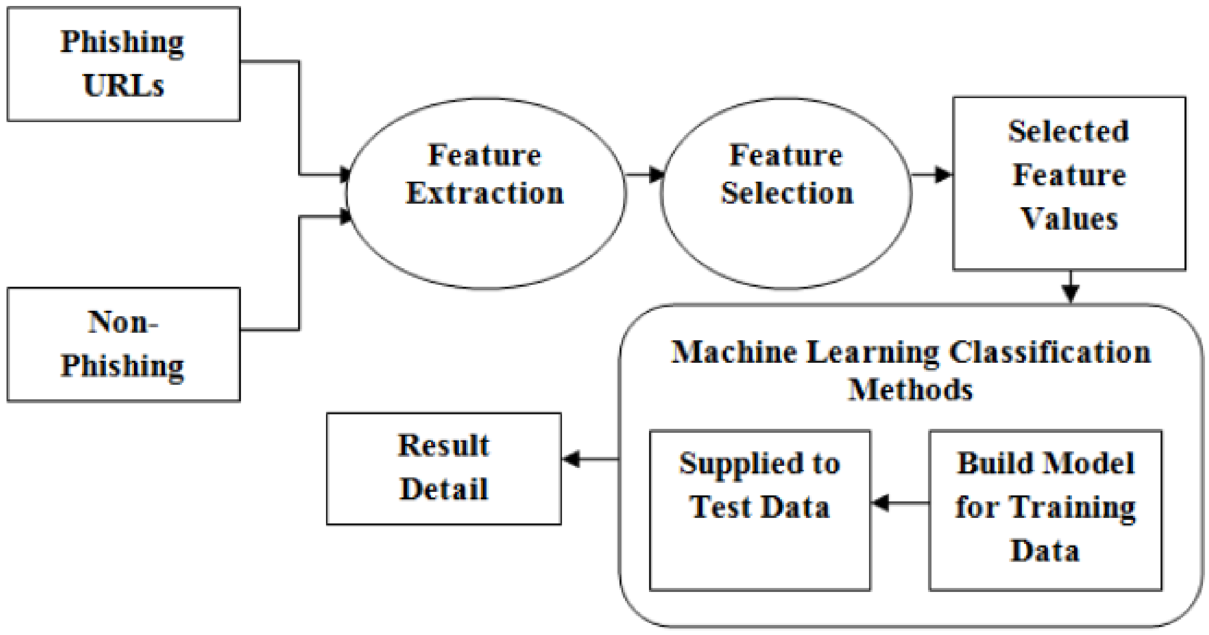


Fig. 4.5 Feature Extraction Diagram

4.6 Preprocessing:

- Normalize and preprocess the extracted features to ensure uniformity and compatibility for the machine learning model.

4.7 Model Training:

- Utilize machine learning algorithms such as Random Forest, Gradient Boosting, or deep learning architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).
- Train the model on the preprocessed dataset, using labeled data to teach the model to distinguish between phishing and legitimate domains.

4.8 Model Evaluation:

- Assess the performance of the trained model using metrics like accuracy, precision, recall, and F1-score.
- Employ techniques like cross-validation or holdout validation to ensure the

model's generalization ability.

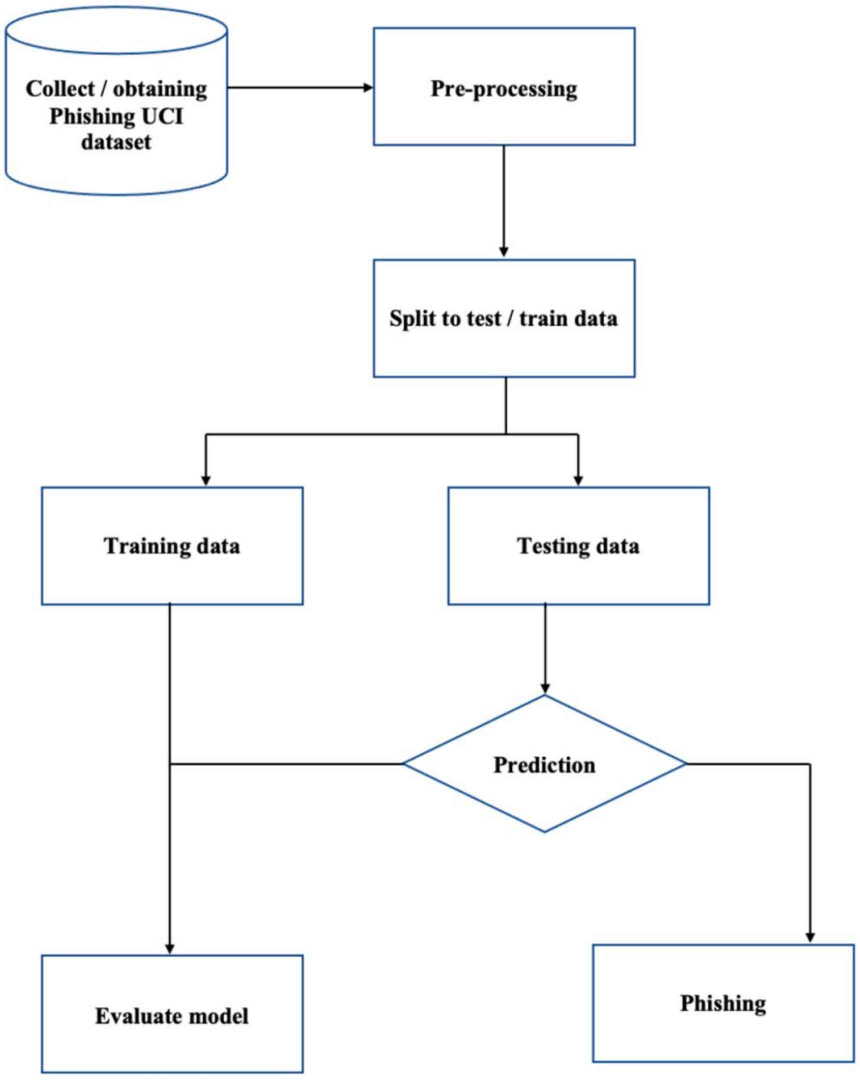


Fig. 4.8 Training Model Evaluation

4.9 Deployment:

- Deploy the trained model as a web service or API accessible to users or integrate it directly into security products and services.
- Ensure scalability and reliability of the deployed system to handle real-time domain checks.

4.10 Continuous Learning and Updating:

- Periodically retrain the model using updated datasets to adapt to emerging phishing techniques and patterns.
- Integrate feedback mechanisms to incorporate user reports and new phishing instances into the training data.

4.11 Integration with Security Tools:

- Integrate the phishing domain detection system with web browsers, email clients, and network security appliances to provide real-time protection for end-users.
- Collaborate with cybersecurity companies and organizations to share threat intelligence and improve the effectiveness of the system.

4.12 Monitoring and Alerting:

- Implement monitoring mechanisms to track the performance of the system and detect any anomalies or degradation in detection accuracy.
- Set up alerting systems to notify administrators or security teams of potential phishing threats identified by the system.

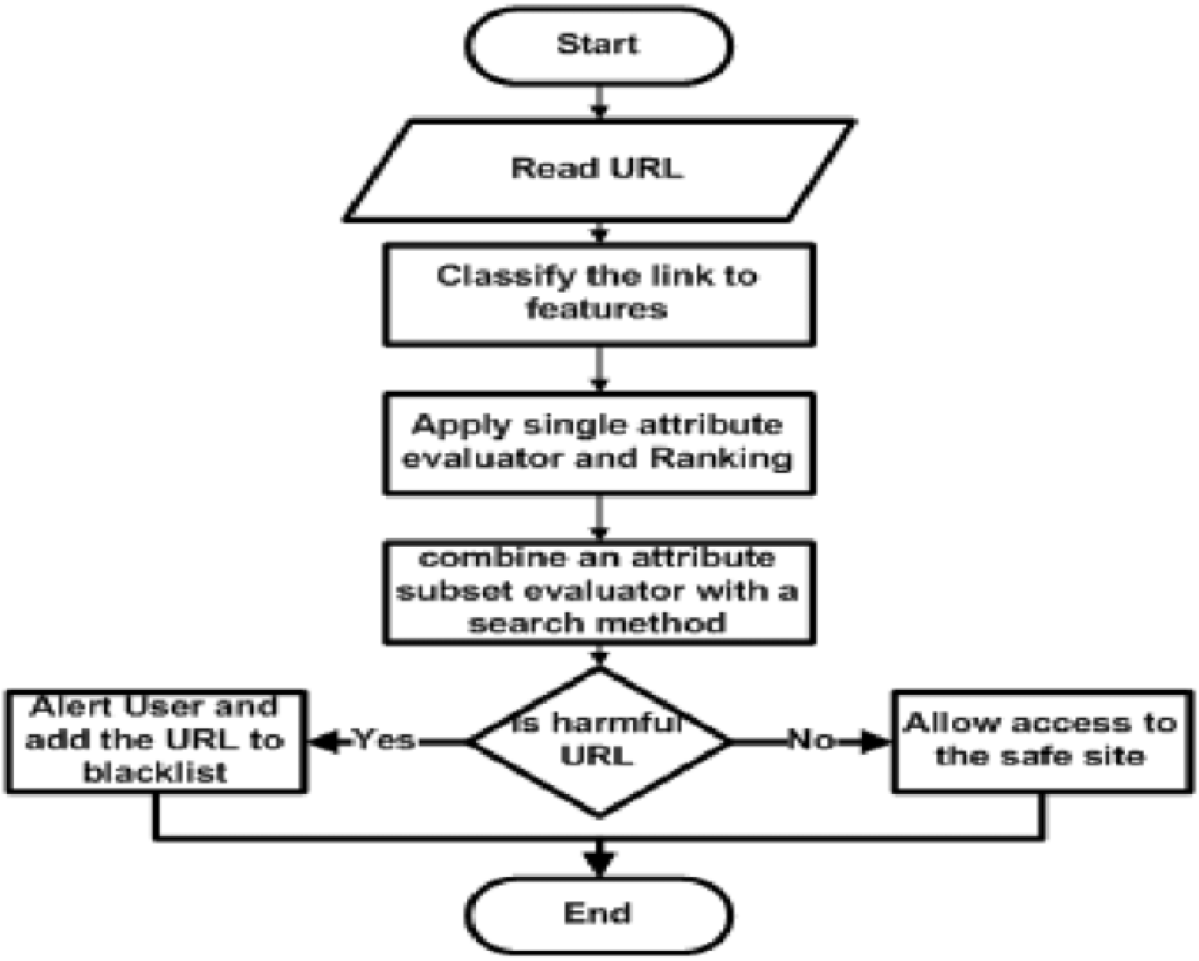

Start
Read URL
Classify the link to features
Apply single attribute evaluator and Ranking
combine an attribute subset evaluator with a search method
Is harmful URL
Yes
No
Alert User and add the URL to blacklist
Allow access to the safe site
End

Fig. 4.12 URL Alert Diagram

4.13 Feedback Loop

- Establish a feedback loop to collect data on false positives and false negatives encountered in real-world usage.
- Use this feedback to fine-tune the model and improve its accuracy and reliability over time.

CHAPTER 5
SYSTEM SPECIFICATION

5.1 User Interface:

- Develop a user-friendly web interface where users can register/login and the give input URLs to be checked.
- Include a form field for users to enter the URL.
- Design the interface to display the result indicating whether the URL is phishing or genuine.
- An interface for uploading phishing URLs by the user who have acknowledgement about URLs is phishing or genuine and user can dispute if model is predicted false result.
- Dataset prepared by admin and uploaded by user/victim can be visible by other user/victim.

5.2 Backend Processing:

- Implement backend logic to register/login and receive URL inputs from the frontend.
- Preprocess the URL data to extract relevant features.
- Integrate the machine learning model for classification. This model should be capable of predicting whether a given URL is phishing or genuine.
- Ensure efficient handling of incoming requests and responses.
- Implement backend to receive phishing/genuine URLs by users and show to other user using my SQL database.

5.3 Machine Learning Model:

- Select a suitable machine learning algorithm for binary classification tasks, such as Random Forest, Logistic Regression, or Gradient Boosting Machines.
- Train the model on a labeled dataset containing URLs labeled as phishing or genuine.
- Tune the hyperparameters of the model to optimize performance, considering metrics like accuracy, precision, recall, and F1-score.

- Serialize the trained model to be used within the web application.

5.4 Data Pipeline:

- Set up a data pipeline to preprocess and transform incoming URLs into features suitable for the machine learning model.
- Handle any necessary data encoding, scaling, or normalization.
- Ensure data integrity and security throughout the pipeline.

5.5 Integration

- Integrate the machine learning model into the backend of the website.
- Establish communication between the frontend and backend components to enable seamless interaction.
- Implement error handling and validation to handle edge cases gracefully.

5.6 Deployment

- Deploy the website and backend system on a web server accessible to users.
- Ensure scalability and reliability of the deployment environment to handle concurrent requests.
- Set up monitoring and logging to track system performance and detect any issues.

5.7 Security Considerations

- Implement security measures to protect the system from common web vulnerabilities such as cross-site scripting (XSS) and SQL injection.
- Apply appropriate access controls and encryption techniques to safeguard sensitive data.
- Regularly update dependencies and patches to mitigate security risks.

5.8 Testing and Quality Assurance

- Conduct thorough testing of the website and backend system to ensure functionality and reliability.
- Perform unit tests, integration tests, and end-to-end tests to validate different components.
- Solicit feedback from users and stakeholders to identify areas for improvement.

5.9 Documentation

- Document the system architecture, components, and APIs for future reference and maintenance.
- Provide user documentation and guides to help users understand how to use the website effectively.

5.10 Maintenance and Updates:

- Establish a plan for ongoing maintenance, including regular updates to the machine learning model and software components.
- Monitor system performance and user feedback to identify opportunities for enhancements and bug fixes.

CHAPTER 6

IMPLEMENTATION AND TESTING

IMPLEMENTATION

6.1 Algorithm

Step 1: Get the link to website

Step 2: Register/Login

Step 3: Enter URL to be checked

Step 4: Prediction Phase

Step 5: Display result

If user want to upload phishing/genuine URLs which he already known or want to dispute about false prediction by model and want to see public dataset.

Step 6: Click dispute button

Step 7: Click to dataset

Step 1: Get the link to website

Firstly, the user has to run the module and get the link to website where he can check URLs. If the user has already run the module in his system, he can use the same URL later on.

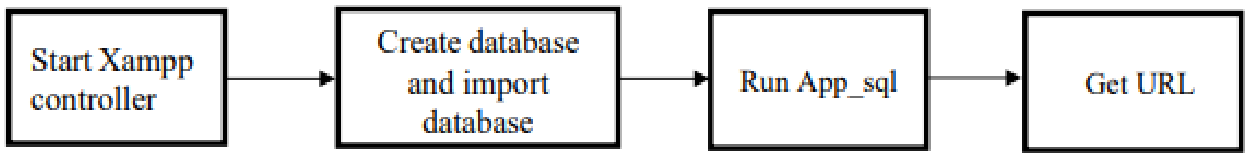


Figure 6.1: steps to get Prediction website URL

Step 2: Register/Login

The user has to paste the link in any browser which results in a website. The user has to register if he's a first-time user or he can login into the website.

Step 3: Enter URL to be checked

After logging in, Enter the URL which has to be checked and click on the predict button.

Step 4: Prediction Phase

Parameters are extracted from the entered URL and a space matrix is created. The space matrix is given as input to the ML model. If the result is -1 the output is given as “Phishing is Detected for this Website”. Otherwise, the output is displayed as “Phishing is not Detected for this Website.”

Step 5: Display Result

The result from the prediction phase is displayed on the front end webpage.

Step 6: Dispute

User can dispute about result and upload true data in public dataset.

Step 7: Public dataset

Public dataset contains list of phishing/genuine URLs created by admin and users/victim.

TESTING

6.2 Methods of Testing

The main aim of the testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operating environments.

In this project, we have developed a GUI and a Machine Learning code which helps in detecting Website URL’s and predicting them phished or not. The main aim of testing this project is to check if the URL is being predicted accurately and check the working performance when different URLs are given as inputs.

The testing steps are:-

1. Unit Testing
2. Integration Testing
3. Validation Testing
4. User Acceptance Testing
5. Output testing

6.3 Test case

Table 1: User Registration Test

Test Case	1
Name of Test	User Registration Test
Input	Email: bindumadhavin2000@gmail.com Password: Bindu@12
Expected Output	Registration Successful
Actual Output	Registration Successful
Result	Successful

Table 2: Login Test

Test Case	1
Name of Test	Login Test
Input	Username: bindumadhavin2000@gmail.com Password: Bindu@12
Expected Output	Login Successful
Actual Output	Login Successful
Result	Successful

Table 3: Parameters Extraction Test

Test Case	1
Name of Test	Extracting Parameters
Input	www.google.com
Expected Output	Parameters are extracted and values are assigned
Actual Output	Parameters are extracted and values are assigned
Result	Successful

Table 4: Prediction Test-1

Test Case	1
Name of Test	Prediction Test
Input	www.google.com
Expected Output	Phishing is not detected for this website
Actual Output	Phishing is not detected for this website
Result	Successful

Table 5: Prediction Test-2

Test Case	2
Name of Test	Prediction Test
Input	Ieeexplore.ieee.org
Expected Output	Phishing is not detected for this website
Actual Output	Phishing is not detected for this website
Result	Successful

Table 6: Prediction Test-3

Test Case	3
Name of Test	Prediction Test
Input	sci-hub.se
Expected Output	Phishing is detected for this website
Actual Output	Phishing is detected for this website
Result	Successful

CHAPTER 7
CONCLUSION AND FUTURE WORKS

7.1 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 web is dependable now. It is conceivable to utilize AI to get information and assemble extraordinary information items. A lay person, completely unconscious 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 Anaconda IDE and was written in Python. The proposed method used four machine learning classifiers to achieve this and a comparative study of the four algorithms was made. A good accuracy score was also achieved. The four algorithms used are K-Nearest neighbor, Kernel Support Vector Machine, Decision Tree and Random Forest Classifier. All the four classifiers gave promising results with the best being Random Forest Classifier with an accuracy score of 96.82%. This model can be deployed in real time to detect the URLs as phishing or legitimate.

7.2 Future Works

Further work can be done to enhance the model by using ensembling models to get 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.

APPENDIX – I
CODING

Program

App.py

```
cursor.execute("SELECT username FROM users WHERE username = %s AND password = %s", (username, password))

        user_data = cursor.fetchone()
from flask import Flask, render_template, request, redirect, url_for, session
import pymysql
from flask_mysqldb import MySQL
import pandas as pd
from sklearn import metrics
import warnings
import pickle
from feature import FeatureExtraction
import numpy as np

app = Flask(_name_)
app.secret_key = 'ashok@123'

warnings.filterwarnings('ignore')

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

```
# Configure MySQL
# app.config['MYSQL_HOST'] = 'localhost'
# app.config['MYSQL_USER'] = 'root'
# app.config['MYSQL_PASSWORD'] = ''
# app.config['MYSQL_DB'] = 'test'

# mysql = MySQL(app).

DB_HOST = 'localhost'
DB_USER = 'root'
DB_PASSWORD = ''
DB_NAME = 'test'
username1=""

def connect_to_database():
    return pymysql.connect(host=DB_HOST,
                           user=DB_USER,
                           password=DB_PASSWORD,
                           database=DB_NAME,
                           cursorclass=pymysql.cursors.DictCursor)

# Routes
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/register', methods=['GET', 'POST'])
```

```
def register():
    if request.method == 'POST':
        # Get form data
        userDetails = request.form
        username = userDetails['username']
        password = userDetails['password']
        email = userDetails['email']
        mobile = userDetails['mobile']
        # Create cursor
        connection = connect_to_database()
        cursor = connection.cursor()
        # Execute query
        cursor.execute("INSERT INTO users(username,email,mobile, password)
VALUES(%s, %s,%s,%s)", (username,email,mobile, password))
        # Commit to DB
        connection.commit()
        # Close connection
        cursor.close()
        return render_template('login.html')
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        # Get form data
        userDetails = request.form
        username = userDetails['username']
        password = userDetails['password']
        # Create cursor
        connection = connect_to_database()
        cursor = connection.cursor()
        # Execute query
        result =  cursor.execute("SELECT * FROM users WHERE username = %s
AND password = %s", (username, password))

        username1 = user_data['username'] if user_data else None
        session['username'] = username1

        if result > 0:
            return render_template('index.html', username=username1)
        else:
            return 'sorry ! something wrong'
        # Close connection
        cursor.close()
    return render_template('login.html')


@app.route('/logout')
def logout():
    session.pop('username', None)  # Remove 'username' from session
    return redirect(url_for('home'))
```

```
@app.route('/index', methods=["GET", "POST"])
def index():
    if 'username' in session:
       username1 = session['username']
       if request.method == "POST":
           url = request.form["url"]
           obj = FeatureExtraction(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]

           pred = "It is {0:.2f} %s safe to go ".format(y_pro_phishing*100)

           return                              render_template('index.html',username=username1,
xx=round(y_pro_non_phishing,2), url=url)
    return render_template("index.html", xx=-1)

@app.route('/phishing',methods=['GET','POST'])
def phishing():
    if request.method == 'POST':
        # Get form data
        userDetails = request.form
        phishing = userDetails['phishing_url']
        genuine = userDetails['genuine_url']
        # email = userDetails['email']
        # mobile = userDetails['mobile']
        # Create cursor
        connection = connect_to_database()
        cursor = connection.cursor()
        # Execute query
        cursor.execute("INSERT INTO dataset(Phishing_URLs,Genuine_URLs)
VALUES(%s,%s)", (phishing,genuine))
        # Commit to DB
        connection.commit()
        # Close connection
        cursor.close()
        return render_template('phishing.html')
    return render_template('phishing.html')


@app.route('/data',methods=['GET','POST'])
def data():
    connection = connect_to_database()
    cursor = connection.cursor()

    try:
        cursor.execute("SELECT * FROM dataset")  # Replace 'your_table_name'
with your actual table name
        data = cursor.fetchall()  # Fetch all rows from the result
        # Close connection
        connection.close()

        return render_template('data.html', res=data)  # Pass the fetched data to the template for rendering
    except Exception as e:
        return f"An error occurred: {e}"



@app.route('/blog')
def blog():
    return render_template('blog.html')

@app.route('/About')
def About():
    return render_template('About.html')



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

Index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta
   name="description"
   content="This is a machine learning based system, with intention  to identify
phishing url and prevent to be victim of phishing attack."
  />
  <meta
   name="keywords"
   content="python phishing url detection , phishing url detection with machine
learning, cyber security and machine learning"
  />
  <meta name="author" content="ItsRaj" />
  <meta name="date" content="2 March 2024" />

  <!-- BootStrap -->
  <link
   rel="stylesheet"
   href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
  />

  <link href="static/styles.css" rel="stylesheet" />
```

```
}

<title>Phishing URL detection</title>
<style>
  .navs {
    background-color: rgb(51, 53, 53);
    font-size: 20px;
  }
  footer {
    background-color: #0087ac;
    color: #fff;
    text-align: center;
    justify-content: center;
    padding: 5px;
    width: 100%;
    position: fixed;
    bottom: 0;
  }
  footer ul li a {
    color: #fff;
    font-size: 15px;
  }
  .right{
    float: inline-end;
  }
  .row{
    display: flex;
    flex-direction: row;
  }
```

```
   </style>
  </head>

  <body>


       <div class="left"></div>
       <ul class="nav navs">
       <li class="nav-item">
         <a href="{{ url_for('index') }}" class="nav-link text-light">home</a>
       </li>
       <li class="nav-item"><a href="{{ url_for('About') }}" class="nav-link text-
light">About</a></li>
       <li class="nav-item"><a href="{{ url_for('blog') }}" class="nav-link text-
light">Blog</a></li>
       <li class="nav-item">
         <a href="{{ url_for('logout') }}" class="nav-link text-warning">logout</a>
       </li>
       <li> <span
         class="navbar-text text-success rights d-flex justify-content-end align-items-
center"
       >
         {% if username %} Welcome, {{ username }} {% else %}
         <li class="nav-item"><a href="/login" class="nav-link ">Login</a></li>

         {% endif %}
       </span></li>
```

```
    </ul>
  </div>











  <div style="text-align: center; margin-top: 10px; font-size: 40px;" class="text-
warning">
    AI/ML System to Detect Phishing Domains (URLs)
  </div>
  <div class="container">
    <div class="row">
      <div class="form col-md" id="form1">
        <br />
        <form action="{{ url_for('index') }}" 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="btn-lg btn-warning">Check here</button>
        </form>
      </div>

      <div class="col-md" id="form2">
        <br />
        <h6 class="right">
          <a href="{{ url }}" target="_blank" class="btn btn-lg btn-light text-lg text-
  danger">{{ url }}</a>
        </h6>

        <br />
        <h3 id="prediction"></h3>
        <button class="button2" id="button2" role="button" target="_blank">
          This url is block you can't use it
        </button>
        <button
          class="button1"
          id="button1"
          role="button"
          onclick="window.open('{{url}}')"
          target="_blank"
        >
          Safe □Continue
        </button>
      </div>
      <!-- <table>
      <thead>
          <tr>
              <th>Column 1</th>
              <th>Column 2</th>
              <!-- Add more table headers as needed -->
          </tr>
      </thead>
      <tbody>
          {% for row in data %}
              <tr>
                  <td>{{ row['column1'] }}</td>
                  <td>{{ row['column2'] }}</td>
                  <!-- Add more table cells for additional columns -->
              </tr>
          {% endfor %}
      </tbody>
  </table> -->
  </div>
  <br />
  <h3 style="text-align: center; margin-bottom: 5px" class="text-danger">
    Be Safe From Phishing Attack □
  </h3>
  <div class="row">
    <div class="col">
  <div class="card" style="width: 300px;">
    <div class="card-title alert alert-success" >Dispute</div>
    <div class="card-body">
```

```
        <div class="card-text alert alert-primary" >
          if prediction is wrong then please upload correctly url is  phishing or
legitimate  !
        </div>
        <a href="{{ url_for('phishing') }}" type="button" class="btn btn-
success">click here</a>
      </div>
    </div>
  </div>
      <div class="col">
        <div class="card" style="width: 300px;">
          <div class="card-title alert alert-success" >Public dataset</div>
          <div class="card-body">
            <div class="card-text alert alert-primary" >
              see realtime phishing and legitimate url dataset upload by users and
admin !
            </div>
            <a href="{{ url_for('data') }}" type="button" class="btn btn-
success">click here</a>

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

```
    <div class="footer">
      <footer>
        <div class="row">
          <div class="col">
            <ul class="nav">
              <li class="nav-item">
                <a href="" class="nav-link">& datasciencetech</a>
              </li>
              <li class="nav-item"><a href="" class="nav-link">About Us</a></li>
              <li class="nav-item">
                <a href="" class="nav-link">Contact Us</a>
              </li>
            </ul>
          </div>
        </div>
      </footer>
    </div>

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

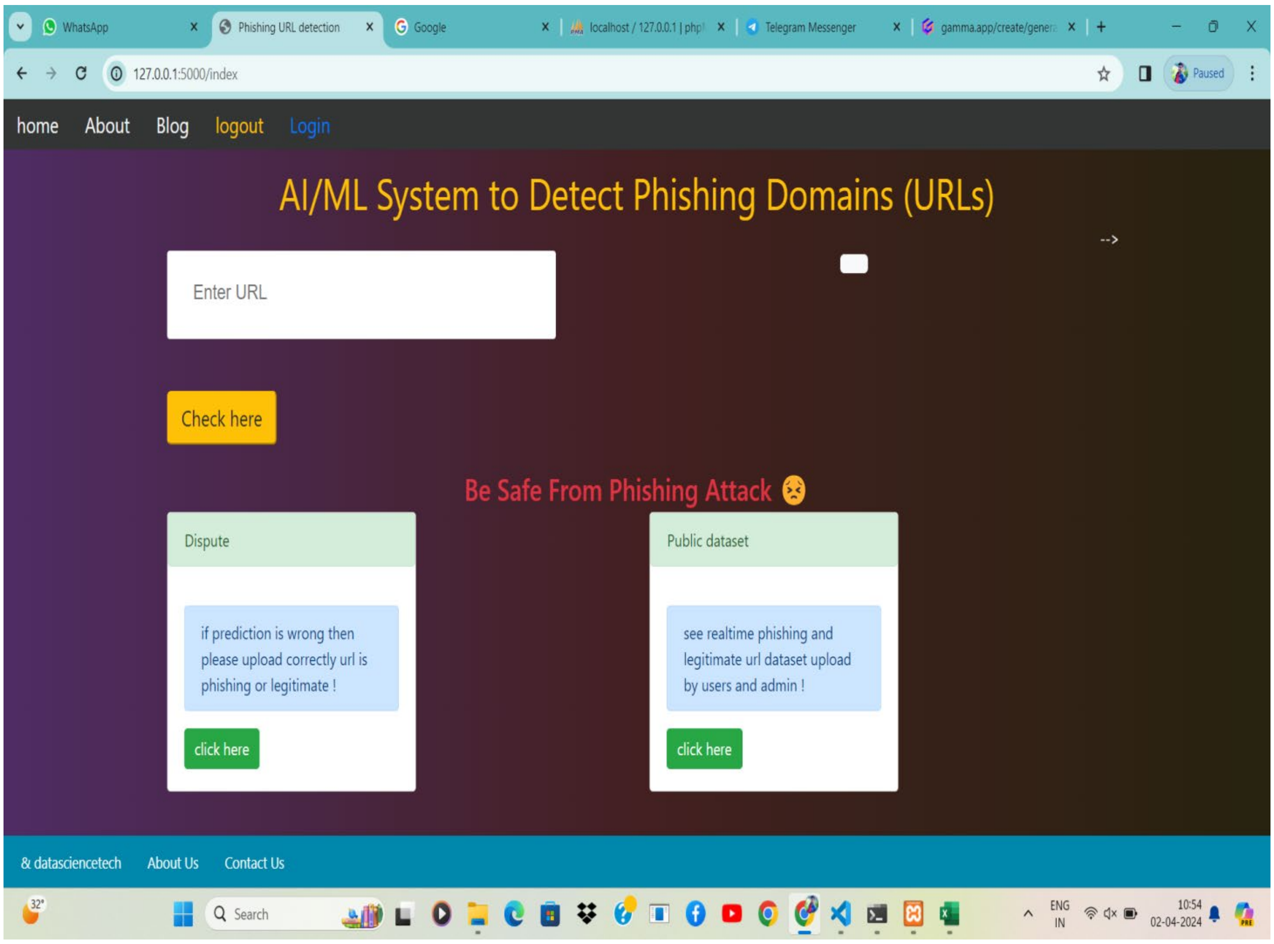
```
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooA
o"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
      integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0J
KI"
      crossorigin="anonymous"
    ></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 + "% genuine , it can be use ...";
        document.getElementById("prediction").innerHTML = label;
        document.getElementById("button1").style.display = "block";
      } else if (0 <= x && x < 0.50) {
        var label = "Website is " + txtx + "% phishing don't use...";
        document.getElementById("prediction").innerHTML = label;
        document.getElementById("button2").style.display = "block";
      }
    </script>
  </body>
</html>
```

APPENDIX -II

OUTPUT

Home Page



AI/ML System to Detect Phishing Domains (URLs)

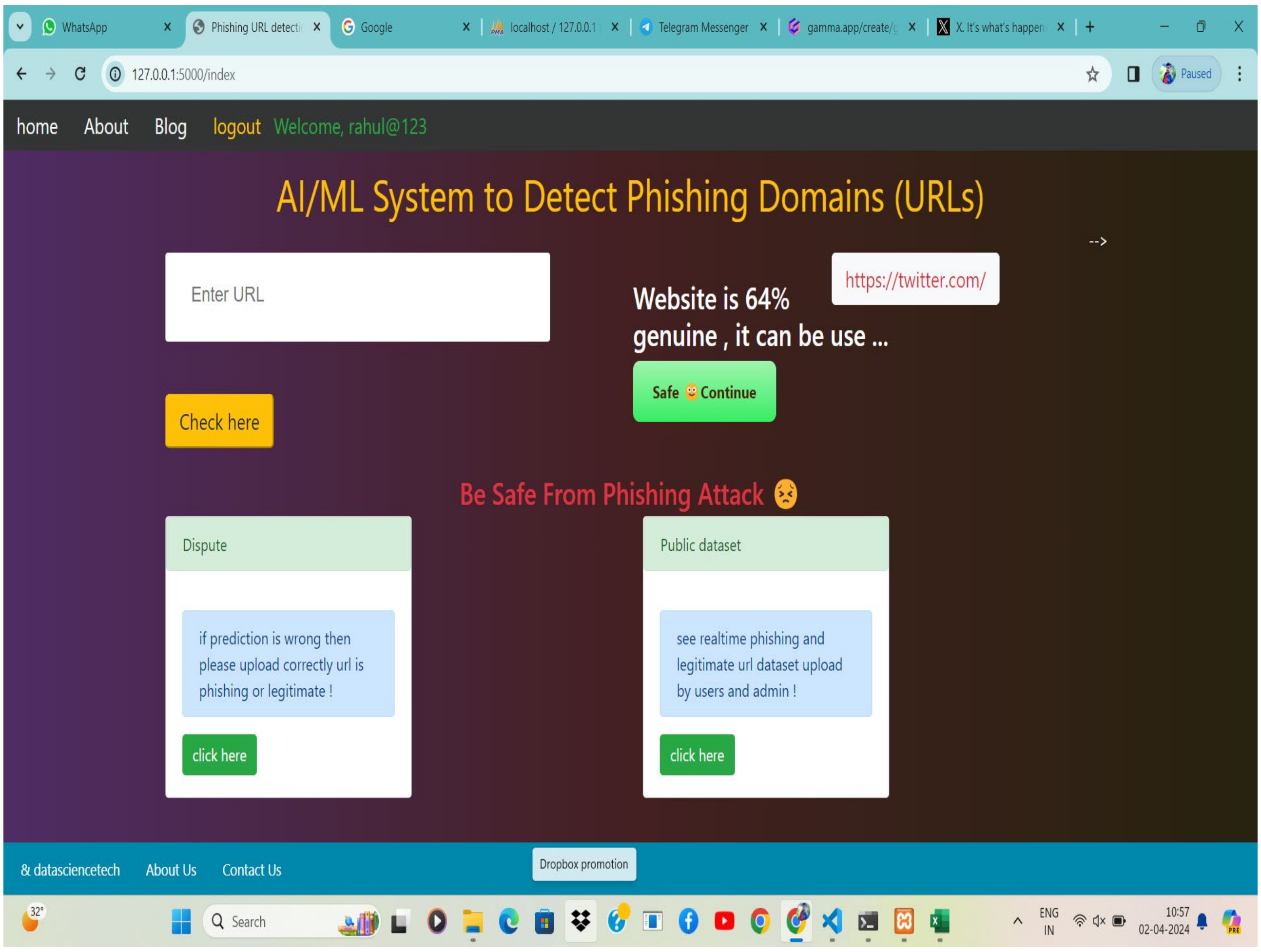
-->

Be Safe From Phishing Attack 😣

if prediction is wrong then please upload correctly url is phishing or legitimate !

see realtime phishing and legitimate url dataset upload by users and admin !

Genuine Page



home About Blog logout Welcome, rahul@123

AI/ML System to Detect Phishing Domains (URLs)

Enter URL

Check here

https://twitter.com/

Website is 64% genuine , it can be use ...

Safe 😊 Continue

Be Safe From Phishing Attack 😣

Dispute

if prediction is wrong then please upload correctly url is phishing or legitimate !

click here

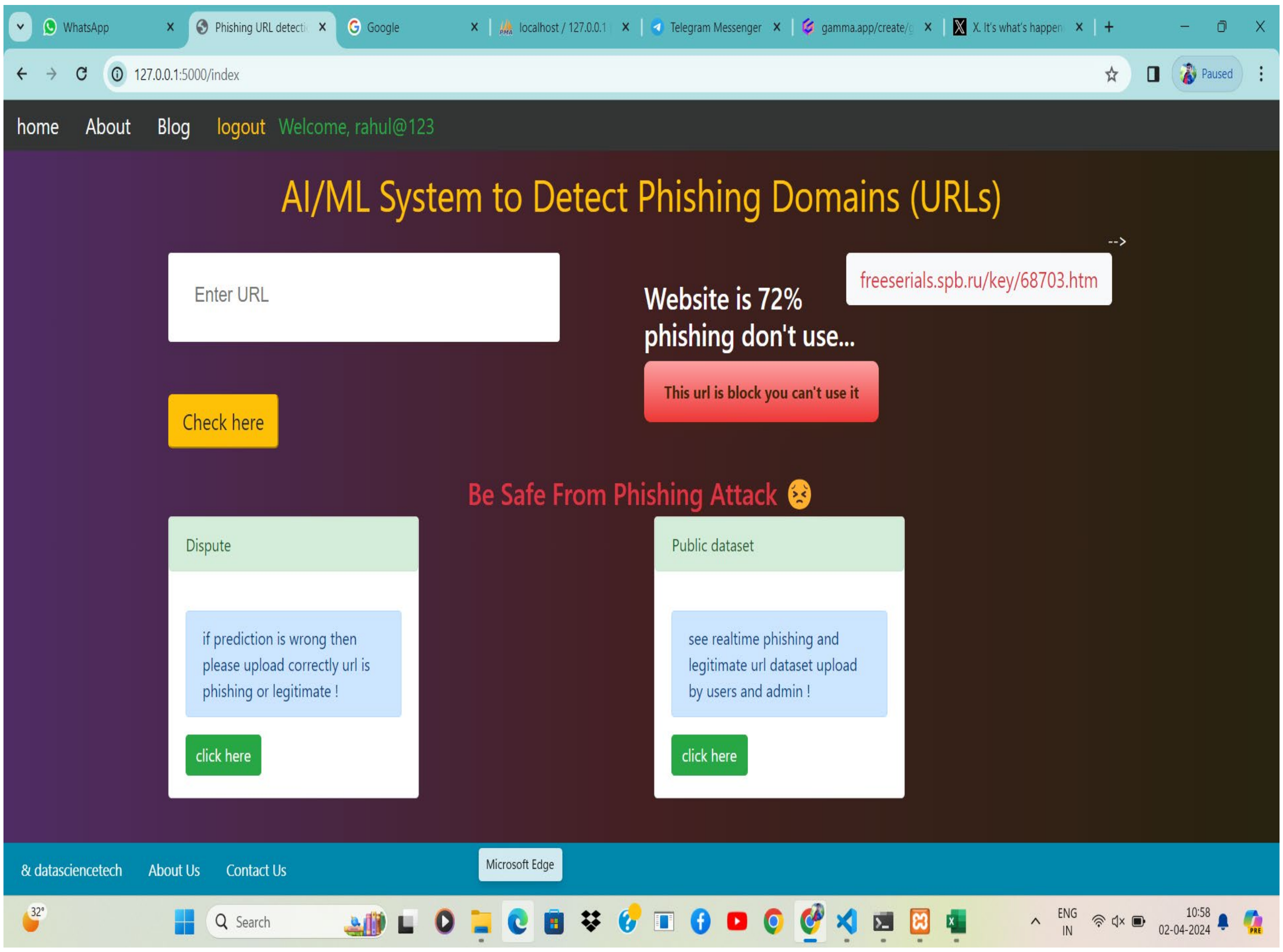
Public dataset

see realtime phishing and legitimate url dataset upload by users and admin !

click here

& datasciencetech About Us Contact Us Dropbox promotion

Phishing URL Page



AI/ML System to Detect Phishing Domains (URLs)

-->

Enter URL

freeserials.spb.ru/key/68703.htm

Website is 72% phishing don't use...

This url is block you can't use it

Check here

Be Safe From Phishing Attack

Dispute

if prediction is wrong then please upload correctly url is phishing or legitimate !

click here

Public dataset

see realtime phishing and legitimate url dataset upload by users and admin !

click here

REFERENCE

- Pawan Prakash; Manish Kumar; Ramana Rao Kompella; Date of Conference: 14-19 March 2010, DOI : 10.1109/INFCOM.2010.5462216

- Amani Alswailem; Bashayr Alabdullah; Norah Alrumayh; Aram Alsedrani; Published in: 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Date of Conference: 01-03 May 2019, DOI: 10.1109/CAIS.2019.8769571

- Lizhen Tang, Qusay H. Mahmoud; Submission received: 7 July 2021 / Revised: 16 August 2021 / Accepted: 17 August 2021 / Published: 20 August 2021, https://doi.org/10.3390/make3030034

- Yuxiang Guan,Futai Zou,Yao Yao,Wei Wang,and Ting Zhu, | Article ID 4678746, Received 02 Jun 2018, Revised 01 Aug 2018, Accepted 02 Sept 2018, Published 26 Sept 2018, https://doi.org/10.1155/2018/4678746

- Altyeb Taha, Maham Zafar, Submission received: 2 October 2021 / Revised: 29 October 2021 / Accepted: 3 November 2021 / Published: 4 November 2021, https://doi.org/10.3390/math9212799

- Zhengyang Fan, Wanru Li, Kathryn Blackmond Laskey and Kuo-Chu Chang, Submission received: 28 November 2023 / Revised: 28 December 2023 / Accepted: 16 January 2024 / Published: 17 January 2024, https://doi.org/10.3390/fi16010031

- Ekaterina Khramtsova; Christian Hammerschmidt; Sofian Lagraa; Radu State; Published in: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Date of Conference: 29 November 2020 - 01 December 2020, DOI: 10.1109/ICDCS47774.2020.00171

- Sami Smadi, Nauman Aslam, Li Zhang, Received 12 April 2017, Revised 18 October 2017, Accepted 5 January 2018, Available online 17 January 2018, Version of Record 6 March 2018, https://doi.org/10.1016/j.dss.2018.01.001