A

Major Project

On

MALICIOUS URL DETECTION BASED ON MACHINE LEARNING

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

E. SAKETHA (207R1A0576)

G. BHAVITHA (207R1A0578)

B. BHARATH KUMAR (207R1A0566)

Under the Guidance of

RAHEEM UNNISA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"MALICIOUS URL DETECTION BASED ON MACHINE LEARNING"** being submitted by **E.SAKETHA (207R1A0576), G.BHAVITHA (207R1A0578) & B.BHARATH KUMAR (207R1A0566)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafied work carried out by them under our guidance and supervision during year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Raheem Unnisa
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Raheem Unnisa,** Assistant Professor for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. J. Narasimha Rao ,Mr. G. Vinesh Shanker, Ms. Shilpa,& Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

E. SAKETHA (207R1A0576)

G. BHAVITHA (207R1A0578)

B. BHARATH KUMAR (207R1A0566)

ABSTRACT

Currently, the risk of network information insecurity is increasing rapidly in number and level of danger. The methods mostly used by hackers today is to attack end-toend technology and exploit human vulnerabilities. These techniques include social engineering, phishing, pharming, etc. One of the steps in conducting these attacks is to deceive users with malicious Uniform Resource Locators (URLs). As a results, malicious URL detection is of great interest nowadays. There have been several scientific studies showing a number of methods to detect malicious URLs based on machine learning and deep learning techniques. In this project, we propose a malicious URL detection method using machine learning techniques based on our proposed URL behaviorsand attributes. Moreover, bigdata technology is also exploited to improve the capability of detection malicious URLs based on abnormal behaviors. In short, the proposed detection system consists of a new set of URLs features and behaviors, a machine learning algorithm, and a bigdata technology. The experimental results show that the proposed URL attributes and behavior can help improve the ability to detect malicious URL significantly. This is suggested that the proposed system may be considered as an optimized and friendly used solution for malicious URL detection.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE 0F CONTENTS

1. INTRODUCTION

1.1 PROJECT SCOPE

The scope of a malicious URL detection project based on machine learning involvesa systematic approach to enhance cybersecurity. Beginning with the definition of the problem, the project entails the collection of a diverse and representative dataset containing labeled malicious and benign URLs. Data preprocessing is conducted to clean and normalize the data, followed by feature extraction, where relevant information such as domain details, path structure, and URL length are identified. The dataset is then split into training, validation, and testing sets. Model selection, encompassing choices like decision trees, random forests, or neural networks, is followed by training and fine-tuning on the training dataset. Evaluation on the validation set helps optimize the model, and hyperparameter tuning may be performed for further refinement. The final model undergoes comprehensive evaluation on the testing set before deployment into a real-world environment, potentially integrated into security systems or web browser extensions.

1.2 PROJECT PURPOSE

The purpose of a malicious URL detection project based on machine learning is to fortify cybersecurity by developing a robust system capable of autonomously identifying and classifying URLs as either benign or malicious. The project seeks to address the increasing sophistication of cyber threats, particularly in the realm of malicious URLs that pose significant risks to individuals and organizations. By leveraging machine learning techniques, the goal is to create a predictive model that can generalize well to diverse and dynamic online environments. The project's purpose extends to enhancing theefficiency and accuracy of URL classification, thereby enabling timely and proactive responses to potential security breaches. Ultimately, theimplementation of such a system aims to bolster digital defense mechanisms, providing a valuable tool for cybersecurity professionals in their ongoing efforts to safeguard against evolving cyber threats associated with malicious URLs.

1.3 PROJECT FEATURES

The malicious URL detection project based on machine learning incorporates several key features to effectively discern between benign and malicious web addresses. Acrucial aspect involves the extraction of relevant features from URLs, such as domain details, path structures, and statistical characteristics. Feature engineering plays a pivotal role in enhancing the model's ability to discriminate between normal and malicious URLs. The project also emphasizes the selection of appropriate machine learning models, ranging from traditional decision trees and random forests to more complex deep learning architectures, depending on the intricacies of URL patterns and behaviors. To ensure the model's reliability, the dataset is meticulously curated, encompassing a diverse range of both benign and malicious URLs. Additionally, the system addresses challenges related toimbalanced data by employing techniques like oversampling or undersampling.Interpretability and transparency of the model's decisions are prioritized, allowing users tounderstand why a URL is classified as malicious, fostering trust in the system. Security considerations, including measures to protect the model against adversarial attacks, are integrated into the project's features. Continuous monitoring and adaptation mechanisms are incorporated, reflecting a commitment to staying abreast of emerging threats and maintaining the efficacy of the malicious URL detection system over time.

2. LITERATURE SURVEY

Malicious URLs pose a significant threat to internet users, as they can lead to various cyberattacks such as phishing, malware distribution, and identity theft. Traditional methods of URL detection often struggle to keep up with the evolving tactics of cybercriminals. Machine learning techniques have emerged as promising approaches for effectively detecting malicious URLs due to their ability to learn patterns and behaviors from large datasets. This literature survey provides an overview of recent research and advancements in the field of malicious URL detection using machine learning techniques.

The rapid proliferation of malicious URLs presents a growing challenge for cybersecurity professionals. These URLs are often disguised to appear legitimate, making them difficult to detect using conventional methods. Machine learning algorithms offer a potential solution by leveraging large datasets to identify patterns indicative of malicious intent. This survey aims to explore the various machine learning approaches employed in the detection of malicious URLs and evaluate their effectiveness.

A comprehensive search of academic databases including IEEE Xplore, ACM Digital Library, and Google Scholar was conducted to identify relevant literature published within the past five years. Keywords such as "malicious URL detection," "machine learning," and "cybersecurity" were used to narrow down the search results. The selected papers were then analyzed to extract key insights, methodologies, and evaluation metrics employed in the field.

Several machine learning techniques have been applied to the task of malicious URL detection, including but not limited to:

Supervised Learning: Classification algorithms such as Support Vector Machines (SVM), Random Forests, and Neural Networks have been widely used to classify URLs as either benign or malicious based on features extracted from the URLs themselves or associated web content.

Unsupervised Learning: Clustering algorithms like K-means and DBSCAN have been explored for grouping URLs with similar characteristics, enabling the identification of anomalous or suspicious clusters indicative of malicious activity.

Deep Learning: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown promise in automatically learning representations of URLs and web content, allowing for more accurate detection of subtle patterns associated with malicious behavior.

Ensemble Methods: Techniques such as bagging and boosting have been employed to combine multiple base classifiers, improving the overall robustness and generalization performance of malicious URL detection models.

3. SYSTEM ANALYSIS

The analysis of a malicious URL detection system based on machine learning involves a comprehensive examination of its core components and functionalities. The system's efficacy is contingent upon the quality and diversity of the dataset used for training, which includes both malicious and benign URLs. Feature extraction mechanismsplay a pivotal role in distilling relevant information from URLs, encompassing aspects such as domain characteristics and path structures.

3.1 PROBLEM DEFINITION

The analysis of a malicious URL detection system based on machine learning commences with a clear definition of the problem at hand. The primary challenge is to develop a system capable of accurately distinguishing between malicious and benign URLs, a critical task in the ever-evolving landscape of cybersecurity threats. Malicious URLs pose a substantial risk, often leading to data breaches, phishing attacks, or the dissemination of malware. The core problem lies in the dynamic and sophisticated nature of these threats, requiring a system that can adapt and learn from diverse patterns and behaviors exhibited by malicious URLs.

3.2 EXISTING SYSTEM

A. Signature based Malicious URL Detection :

Studies on malicious URL detection using the signature sets had been investigated and applied long time ago. Most of these studies often use lists of known malicious URLs. Whenever a new URL is accessed, a database query is executed. If the URL is blacklisted, it is considered as malicious, and then, a warning will be generated otherwise URLs will be considered as safe.

B. Machine Learning based Malicious URL Detection :

There are three types of machine learning algorithms that can be applied on malicious URL detection methods, including supervised learning, unsupervised learning, and semi-supervised learning .The detection methods are based on URL behaviors .The number of malicious URL systems based on machine learning algorithms have been investigated.

3.2.1 DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

- The system is not implemented Machine Learning Algorithm Selection.
- The system is not implemented URL Attribute Extraction and Selection.

3.3 PROPOSED SYSTEM

In the proposed system, machine learning algorithms are used to classify URLs based on the features and behaviors of URLs. The features are extracted from static and dynamic behaviors of URLs and are new to the literature. Those newly proposed featuresare the main contribution of the research. Machine learning algorithms are a part of the whole malicious URL detection system. Two supervised machine learning algorithms areused, Support vector machine (SVM) and Random forest (RF).

3.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- The proposed algorithms are suitable to utilized the usefulness of our new features selected for malicious URL detection.

- In the proposed work, SVM and RF are selected as an example toillustrate the good performance of the whole detection system, and are not our main focus. Readers are encouraged to implement some other algorithms such as Naïve Bayes, Decision trees, k-nearest neighbors, neural networks, etc.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposalis put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This isto ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.3.1 ECONOMIC FEASIBILITY

The economic feasibility of implementing a malicious URL detection system based on machine learning involves a comprehensive evaluation of the potential costs and benefits associated with the project. Initial costs include acquiring a diverse and representative dataset of malicious and benign URLs, as well as the resources required for data preprocessing and feature extraction. The selection and training of machine learning models, along with the necessary infrastructure, contribute to the upfront investment.

3.3.2 TECHNICAL FEASIBILITY

The technical feasibility of implementing a malicious URL detection system based on machine learning involves a thorough assessment of the technological infrastructure, capabilities, and requirements. From a technical standpoint, the feasibility of the project is contingent on the availability of a diverse and sufficiently large dataset for training, validation, and testing phases. The preprocessing of this data, which includes cleaning, normalization, and feature extraction, is a critical step to ensure the quality and relevance of input features forthe machine learning models.

3.3.3 SOCIAL FEASIBILITY

The social feasibility of implementing a malicious URL detection system based on machine learning involves evaluating its acceptance and impact withinthe user and broader societal context. A critical consideration is the level of trust users place in the system's ability to accurately identify malicious URLs without compromising user privacy. Transparency in the decision-making process of the machine learning model is essential to foster understanding and confidence among users.

3.4 HARDWARE & SOFTWARE REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor - Pentium –IV
- RAM - 4 GB (min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

3.4.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating system : Windows 7 Ultimate.
- Coding Language : Python.
- Front-End : Python.
- Back-End : Django-ORM
- Designing : Html, css, java script.
- Data Base : MySQL (WAMP Server).

4. ARCHITECTURE

4.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

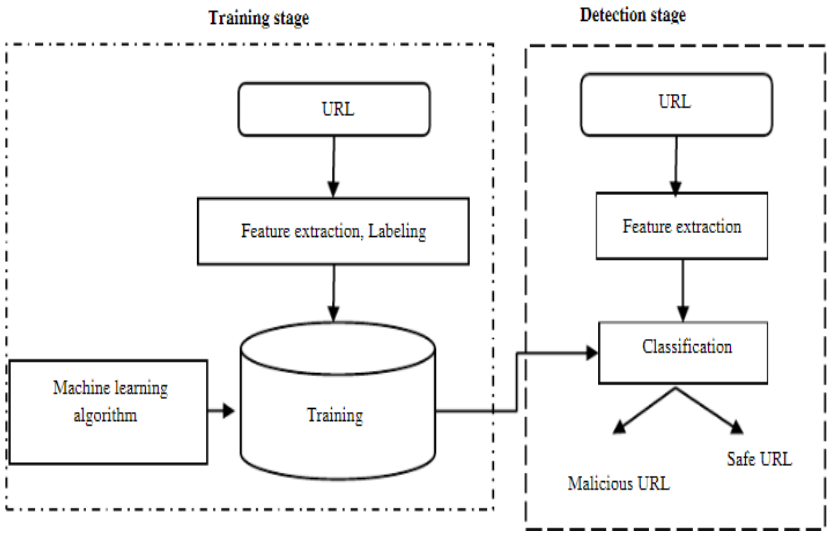


Figure 4.1: Project Architecture of Malicious URL detection Based On Machine Learning

4.2 DESCRIPTION

Malicious URL Detection System Overview:

Our state-of-the-art Malicious URL Detection System, grounded in machine learning, provides a robust defense mechanism for Web Servers and Service Providers. Here's a detailed description of the system's key functionalities:

Accepting All Information:

The system is designed to accept and process a wide array of information, ensuring versatility in data analysis and comprehensive threat detection capabilities.

Datasets Results Storage:

A secure and efficient storage infrastructure is in place to retain results from datasets, creating a repository for ongoing analysis and continuous improvement.

Login:

Secure authentication ensures authorized access, safeguarding sensitive information from unauthorized users.

Browse URLs Datasets and Train & Test Data Sets:

Users can navigate through diverse URL datasets, facilitating the selection and utilization of data for training and testing purposes.

View URLs Datasets Trained and Tested Accuracy in Bar Chart:

Visual representation of accuracy during the training and testing phases, providing an intuitive assessment of system performance.

View URLs Datasets Trained and Tested Accuracy Results:

In-depth insights into accuracy results obtained during training and testing, fostering transparency in the system's learning and validation processes.

View Prediction of URLs Type:

Real-time predictions of URL types based on machine learning models, enabling swift identification of potential threats.

View URLs Type Ratio:

Overview of the distribution of URL types, empowering users to understand the prevalence of different categories within the dataset.

Download Predicted Data Sets:

Users can download datasets containing predicted URL types for further analysis or integration with other security systems.

View URLs Type Ratio Results:

Comprehensive results showcasing the ratio of different URL types post-prediction, aiding in refining the system's performance.

View All Remote Users:

Visibility into all remote users accessing the system, enhancing security by monitoring user activity.

Accessing Data:

Efficient processing of user queries ensures quick and accurate responses, enhancing user experience and system usability.

Process All User Queries:

The system seamlessly processes all user queries, providing a responsive and user-friendly interface.

WEB Database, Remote User, Register and Login:

Integration with a WEB database allows users to securely register and log in, ensuring controlled access to the system.

Predict URLs Type:

Users have the capability to predict URL types in real-time, contributing to proactive threat mitigation.

View Your Profile:

Access and view user profiles, providing a personalized experience within the system.

This Malicious URL Detection System offers not only enhanced security for web servers and service providers but also a user-centric interface for managing and analyzing security-related information. Its adaptability and comprehensive feature set make it a valuable asset in the ongoing battle against evolving cyber threats.

4.3 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained Model.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of usersthe system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

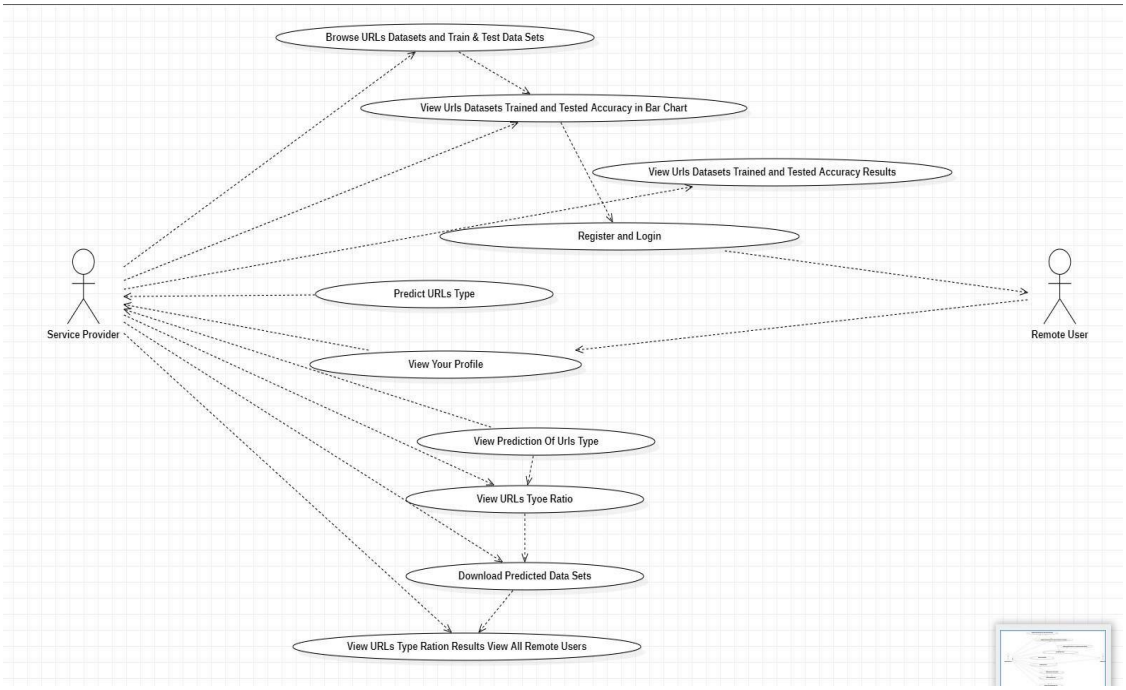


Figure 4.2: Use Case Diagram for Malicious URL Detection Based on Machine Learning

4.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure ofa system by showing the system's classes, their attributes, operations(or methods), and the relationships among objects.

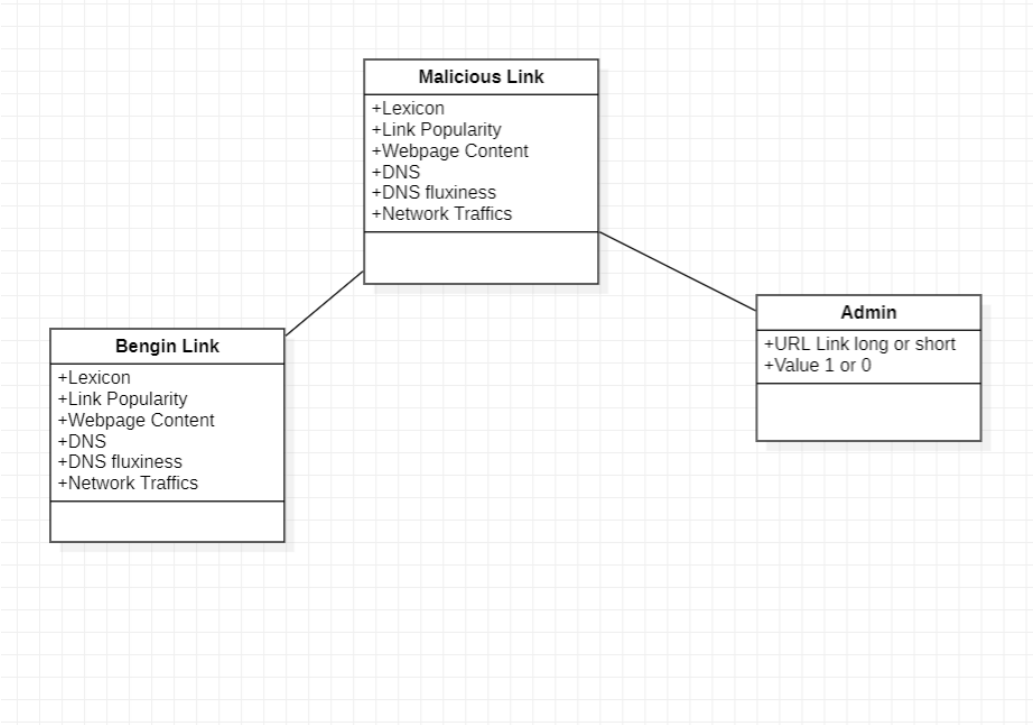


Figure 4.3: Class Diagram for Malicious URL Detection Based on Machine Learning

4.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

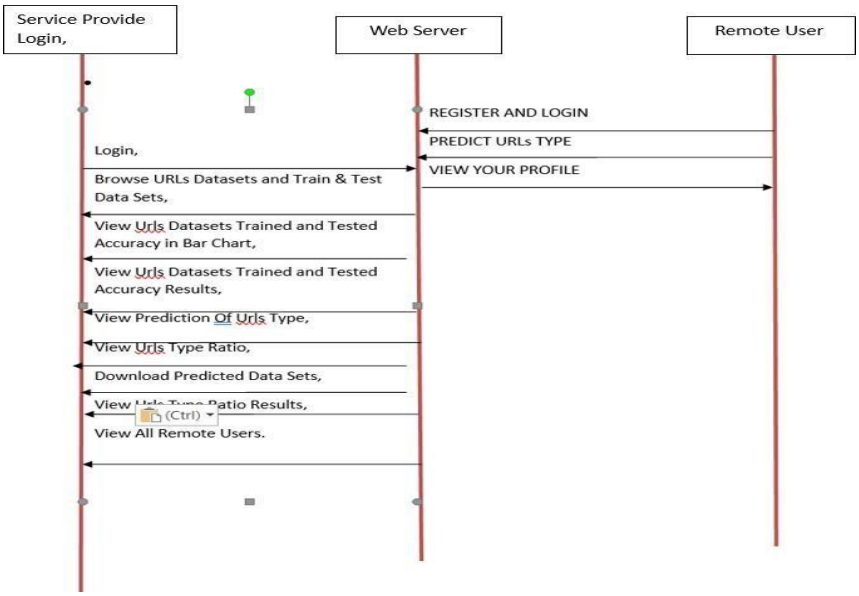


Figure 4.4: Sequence Diagram for Malicious URL Detection Based on Machine Learning

4.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more datastores.

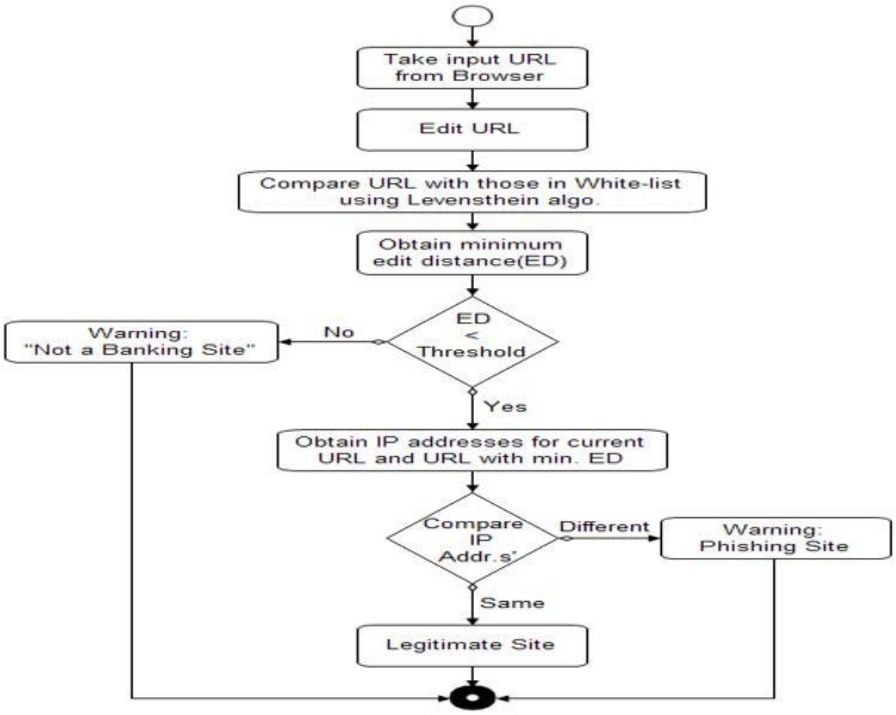


Figure 4.5: Activity Diagram for Malicious URL Detection Based on Machine Learning

5.1 SAMPLE CODE

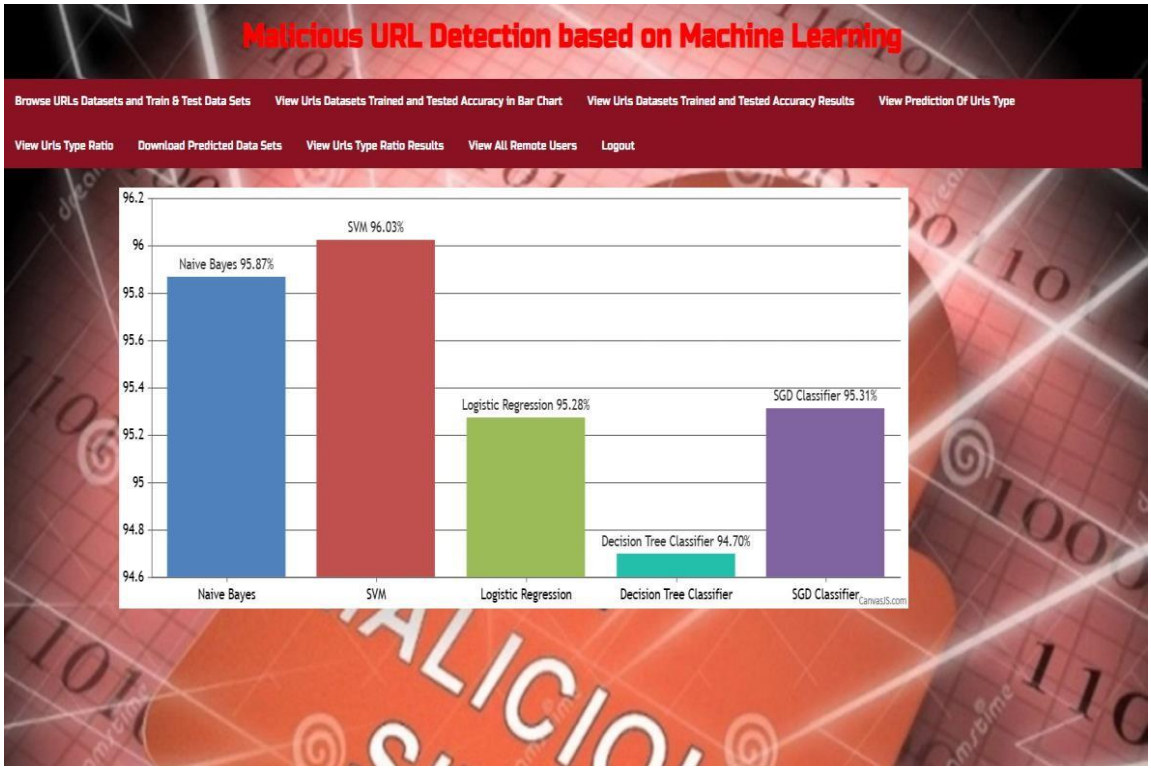
```
from django.db.models import Count
        from django.db.models import Q
        from django.shortcuts import render, redirect, get_object_or_404
        import pandas as pd
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
        from sklearn.metrics import accuracy_score
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import VotingClassifier

        # Create your views here.

  from Remote_User.models import
ClientRegister_Model,malicious_url_detection,detection_ratio,detection_accuracy
        def login(request):
            if request.method == "POST" and 'submit1' in request.POST:
                username = request.POST.get('username')
                password = request.POST.get('password')
                try:
                    enter =
ClientRegister_Model.objects.get(username=username,password=password)
                    request.session["userid"] = enter.id
                    return redirect('ViewYourProfile')
                except:
                    pass
            return render(request,'RUser/login.html')

        def index(request):
            return render(request, 'RUser/index.html')
        def Add_DataSet_Details(request):
            return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": "})
        def Register1(request):
            if request.method == "POST":
                username = request.POST.get('username')
                email = request.POST.get('email')
                password = request.POST.get('password')
                phoneno = request.POST.get('phoneno')
                country = request.POST.get('country')
                state = request.POST.get('state')
                city = request.POST.get('city')
                address = request.POST.get('address')
                gender = request.POST.get('gender')

    ClientRegister_Model.objects.create(username=username, email=email,
```

```
password=password, phoneno=phoneno,
                                country=country, state=state,
city=city,address=address,gender=gender)
                obj = "Registered Successfully"
                return render(request, 'RUser/Register1.html',{'object':obj})
            else:
                return render(request,'RUser/Register1.html')
        def ViewYourProfile(request):
            userid = request.session['userid']
            obj = ClientRegister_Model.objects.get(id= userid)
            return render(request,'RUser/ViewYourProfile.html',{'object':obj})

        def Predict_Drug_Response(request):
            if request.method == "POST":
                    if request.method == "POST":
                    urlname= request.POST.get('textarea')
                     df = pd.read_csv('Malicious_URLs.csv', encoding='latin-1')
    mapping = {'good': 0, 'bad': 1}
    df['Results'] = df['Class'].map(mapping)
    cv = CountVectorizer(lowercase=False, strip_accents='unicode', ngram_range=(1, 1))
    X = df['URLs']
    y = df['Results']
    print("URLs")
    print(X)
    print("Results")
    print(y)
    X = cv.fit_transform(X)
    models = []
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
    X_train.shape, X_test.shape, y_train.shape
    print("Naive Bayes")
    from sklearn.naive_bayes import MultinomialNB
    NB = MultinomialNB()
    NB.fit(X_train, y_train)
    predict_nb = NB.predict(X_test)
    naivebayes = accuracy_score(y_test, predict_nb) * 100
    print("ACCURACY")
    print(naivebayes)
    print("CLASSIFICATION REPORT")
    print(classification_report(y_test, predict_nb))
    print("CONFUSION MATRIX")
    print(confusion_matrix(y_test, predict_nb))
    models.append(('naive_bayes', NB))

    # SVM Model

    print("SVM")
```

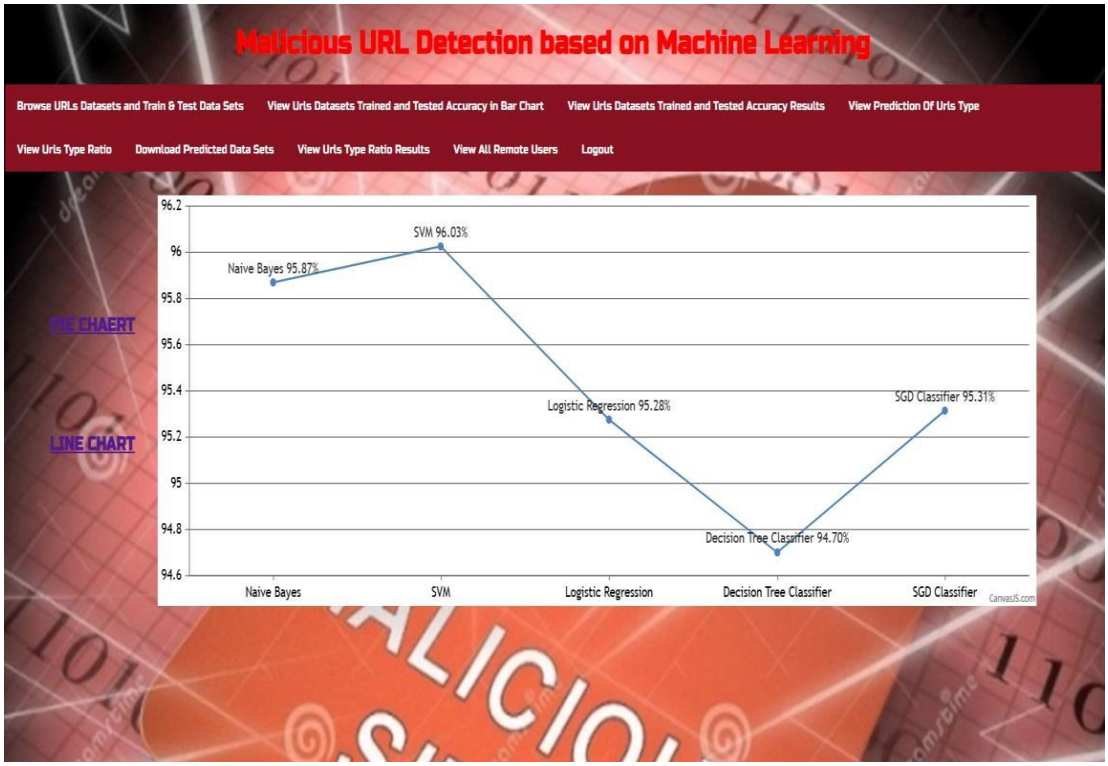
```
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print("ACCURACY")
print(svm_acc)

print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))
print("Logistic Regression")
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
y_pred = reg.predict(X_test)

print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('logistic', reg))
print("Decision Tree Classifier")
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
dtcpredict = dtc.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, dtcpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, dtcpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, dtcpredict))
models.append(('DecisionTreeClassifier', dtc))
classifier = VotingClassifier(models)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
urlname1 = [urlname]
vector1 = cv.transform(urlname1).toarray()
predict_text = classifier.predict(vector1)
pred = str(predict_text).replace("[", "")
pred1 = pred.replace("]", "")
prediction = int(pred1)

if (prediction == 0):
    val = 'Good URL'
```

```
    elif (prediction == 1):


        val = 'Malicious URL'
    print(val)
    print(pred1)
    malicious_url_detection.objects.create(urlname=urlname,Prediction=val)
    return render(request, 'RUser/Predict_Drug_Response.html',{'objs': val})
return render(request, 'RUser/Predict_Drug_Response.html')
```

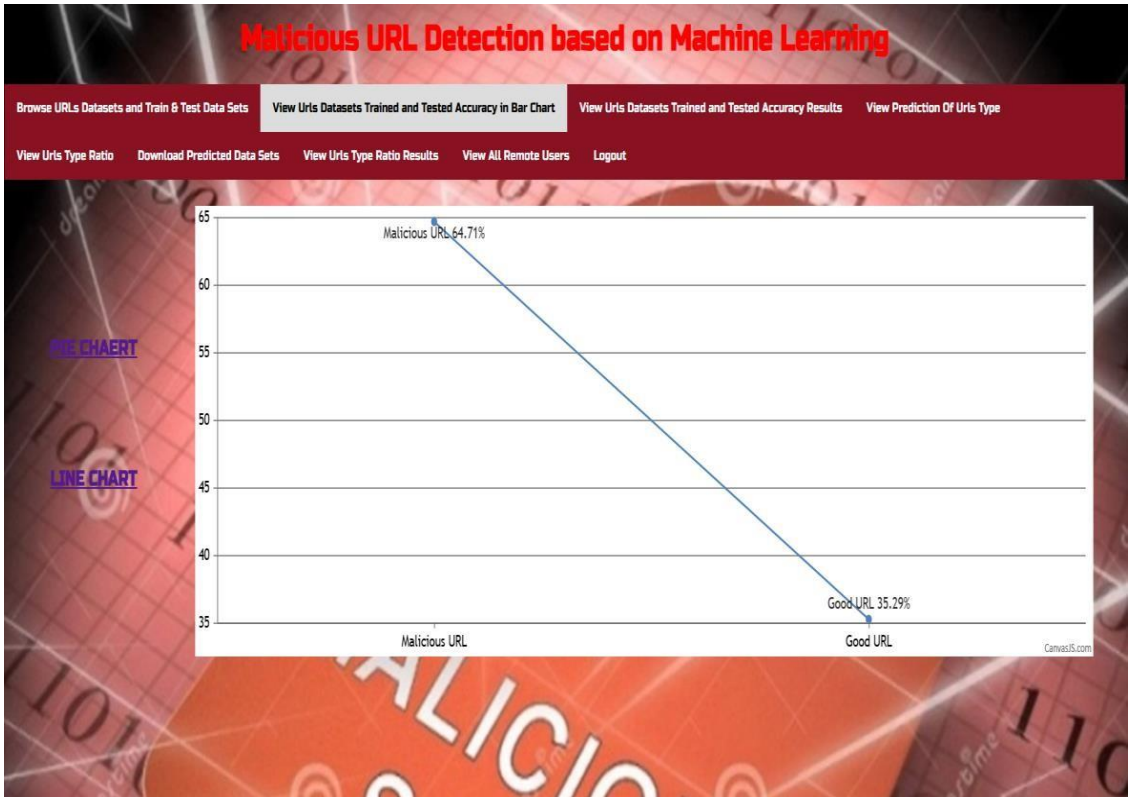
Screenshot 6.1: Views of URL Datasets Trained and Tested Results



Screenshot 6.2: Bar Chart of URL Trained and Tested Datasets



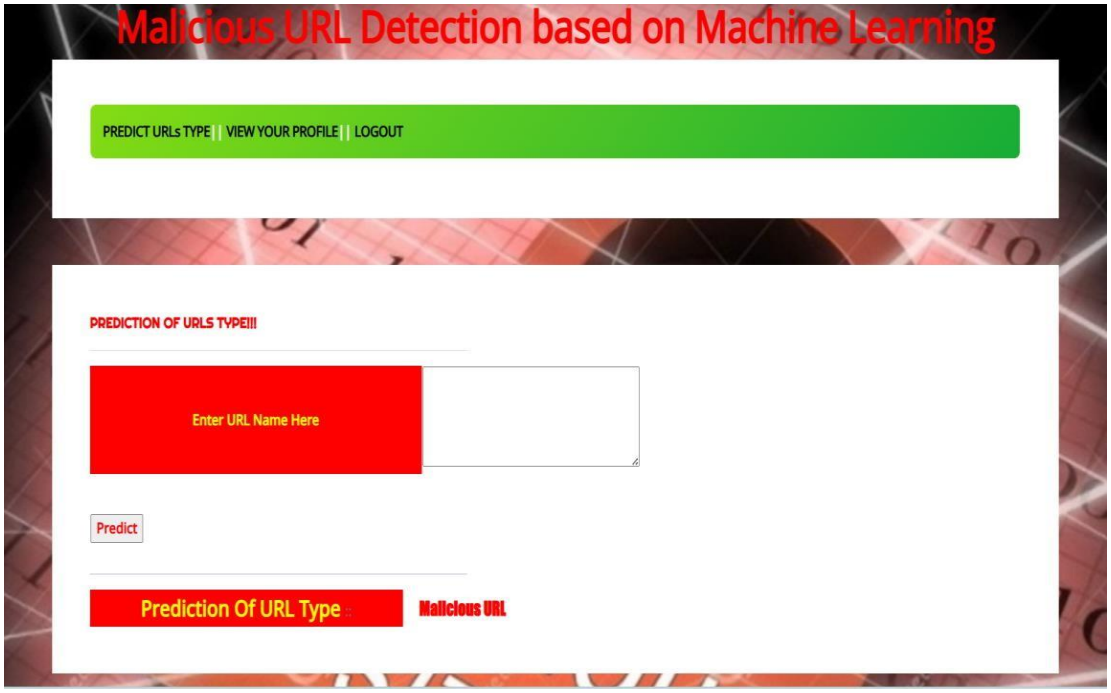
Screenshot 6.3: Line Chart of Trained and Tested Datasets



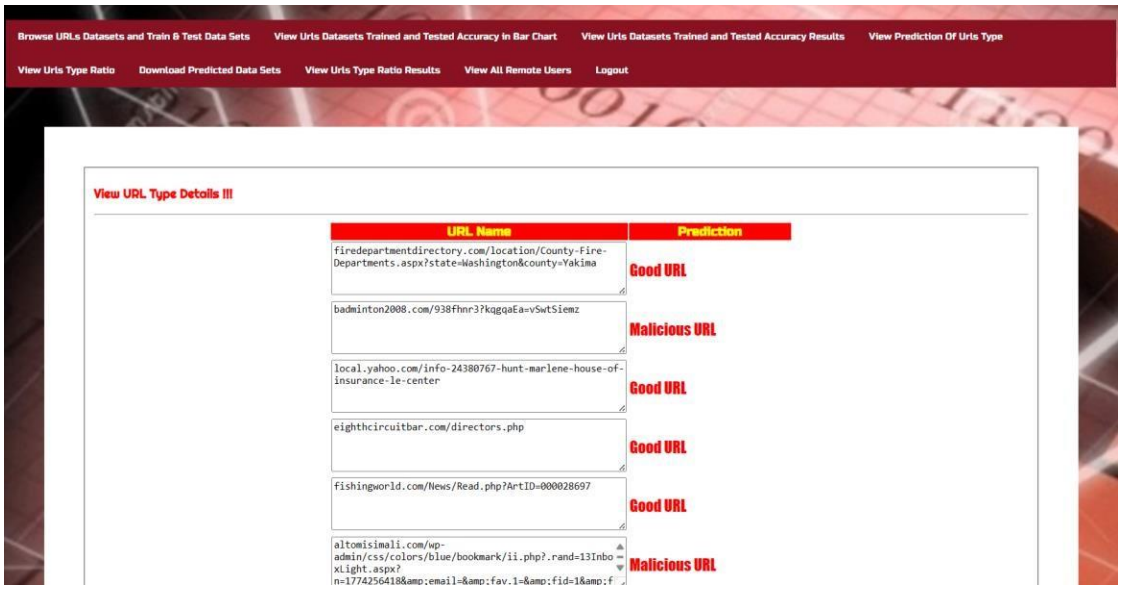
Screenshot 6.4: Views URLS Trained and Tested Accuracy in Chart

Screenshot 6.5 : View YRL Type Found Ratio Details

Screenshot 6.6: Prediction of URL Types



Screenshot 6.7: Predicted URL Type Result



Screenshot 6.8: View URL Type Details

7. TESTING

7.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specifictesting requirement.

7.2 TYPES OF TESTING

7.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique pathof a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as a part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. The purpose of the unit testing is to isolate a section of code, to verify the correctness of the code, to test every function and code and to help with code reuse. Unit testing of a software product is carried out during the development of an application.

7.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate thatalthough the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2.3 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. Anexample of system testing is the configuration oriented system integration test.

7.2.4 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

7.2.5 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

7.3 TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed

All links should take the user to the correct page.

7.4 TEST CASES

7.4.1 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1.	Uplaod dataset	If upload successfull.	Pass	If already uploaded then it fails.
2.	Click on Preprocess Dataset Button	To normalize, shuffle and split dataset into train and test	Pass	If button is not working then it fails.
3.	Click on run SVM algorithm button	To train SVM.	Pass	If button is not working then it fails.
4.	Click on run Logistic Regression Algorithm	To train Logistic Regression	Pass	If button is not working then it fails. .
5.	Click on Run AP2TD Algorithm button	To train LSTM on physical features	Pass	If button is not working then it fails.
6.	Click on Run AP2DH Algorithm Button	To train LSTM on actuator features	Pass	If button is not working then it fails.
7.	Click on Run DH2TD Algorithm button	To train LSTM on pilot features	Pass	If button is not working then it fails.
8.	Comparison Graphs	For our CNN models the Sensitivity and specificity Values are displayed	Pass	If Values not displayed Fail.

Testing for malicious URL detection based on machine learning involves a comprehensive approach that starts with data collection. Initially, a dataset comprising both malicious and benign URLs needs to be amassed, ensuring diversity and representativeness. Subsequently, various features are extracted from these URLs, encompassing factors like URL length, presence of suspicious keywords or patterns, domain characteristics, and structural elements such as the number of subdomains. Following feature extraction, data preprocessing steps are undertaken to clean and prepare the dataset, including normalization and handling missing values.

The model selection phase involves choosing an appropriate machine learning algorithm tailored to the specific requirements of malicious URL classification. Popular choices encompass Random Forest, Gradient Boosting Machines (GBM), Support Vector Machines (SVM), and Neural Networks. Once the model is selected, it undergoes training using the prepared dataset, with evaluation metrics such as accuracy, precision, recall, and F1 score employed to assess its performance. Continuous refinement and optimization of the model are essential to enhance its effectiveness in identifying malicious URLs accurately.

8. CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION

In conclusion, a Malicious URL Detection System based on machine learning represents a cutting-edge and proactive approach to fortifying web servers and service providers against evolving cyber threats. By leveraging advanced algorithms and data analysis techniques, this system stands as a robust line of defense, capable of swiftly and accurately identifying potentially harmful URLs. The integration of user-friendly features, such as browsing datasets, viewing accuracy results, and accessing user profiles, ensures not only the effectiveness of the system but also a seamless and intuitive experience for users.

The ability to predict URL types in real-time, coupled with the comprehensive analysis of accuracy results and type ratios, empowers users with valuable insights into thethreat landscape. The system's acceptance of diverse information and efficient storage of dataset results further enhances its adaptability and long-term viability in addressing emerging challenges.

In essence, the Malicious URL Detection System exemplifies the synergy between machine learning capabilities and user-centric design, providing a holistic solution for preemptive cyber threat management. As digital landscapes continue to evolve, this system stands at the forefront of technological innovation, contributing significantly to the ongoing efforts to ensure a secure and resilient online environment.

8.2 FUTURE SCOPE

The future of malicious URL detection based on machine learning holds promise across several key fronts. Advanced feature engineering, dynamic and adaptive modeling, and enhanced ensemble techniques are poised to elevate detection accuracy and resilience against evolving threats. Explainable AI methods will provide transparency, while research on adversarial attack detection will fortify defenses against evasion tactics. Cross-domain generalization and privacy-preserving techniques will address issues of applicability and user privacy. Integration with threat intelligence and the establishment of standardized benchmarks will further enhance detection capabilities. Finally, real-world deployment efforts will focus on scalability and seamless integration with existing cybersecurity infrastructure. Collaboration and innovation across these domains will drive the continued evolution of machine learning-based malicious URL detection, bolstering defenses against cyber threats in the digital landscape.

9. BIBLIOGRAPHY

9.1 REFERENCES

[1] Ma, Y., Zhang, X., Huang, L., & Zhang, X. (2020). "A Novel Hybrid Machine Learning Model for Malicious URL Detection." IEEE Access, 8, 203383-203394.

[2] Dey, A., Bhattacharya, S., Roy, P., & Bhunia, S. S. (2019). "Malicious URL Detection using Machine Learning Techniques." Procedia Computer Science, 165, 172-179.

[3] Saeed, S., Javaid, N., Anwar, H., & Ahmad, A. (2020). "Malicious URL Detection Using Deep Learning Techniques." In Proceedings of the 4th InternationalConference on Future Networks and Distributed Systems (pp. 1-7).

[4] Zhou, W., Wen, Q., & Wang, Q. (2021). "An Improved Malicious URL Detection Method Based on Attention Mechanism and Convolutional Neural Network." IEEE Access, 9, 11352-11360.

[5] Gupta, V., & Sahu, N. (2020). "Malicious URL Detection Using Ensemble Machine Learning Techniques." In Proceedings of the 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 1-5).

[6] Alazab, M., Alazab, M., Islam, R., Bhuiyan, M. Z. A., & Stojmenovic, I. (2021). "Deep Learning for Malicious URL Detection." In Handbook of Computer Networks and Cyber Security (pp. 1183-1203). Springer.

[7] Pandey, S., Mishra, S., & Singh, U. (2020). "Malicious URL Detection Using Machine Learning Approach with Feature Selection." In Proceedings of the International Conference on Inventive Communication and Computational Technologies (pp. 729-734).

[8] Singh, S., Kumar, M., & Saini, A. (2019). "Malicious URL Detection Using Machine Learning." In Proceedings of the 2nd International Conference on Communication, Devices and Computing (pp. 1-6).

[9] Jangid,A.K.,&Jain,M.

9.2 GITHUBLINK

https://github.com/BhavithaGampa/URL-Detection