A PROJECT REPORT
ON

ANONYMITY AND CONFIDENTIALITY IN WEBSITE USING ML

BY

ADITYA MISHRA (2101321530003)

ANSHUMAN SONI (2101321530007)

ANUJ MISHRA (2101321530008)

SHIVAM SINGH (2101321530045)

UNDER THE GUIDANCE OF

Mr. UMASHANKER SHARMA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING),

GREATER NOIDA INSTITUTE OF TECHNOLOGY ENGG.INSTITUTE, GREATER NOIDA

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

May, 2025

Department of CSE (AI & ML)

Session 2024-2025

Project Completion Certificate

Date: 04/05/2025

This is to certify that **Mr. Aditya Mishra** bearing Roll No.2101321530003, student of 4th year CSE-AIML has completed project program (KCS-851) with the Department of CSE-AIML from 26-Feb-25 to 30-May-25.

He worked on the Project Titled **"ANONYMITY AND CONFIDENTIALITY IN WEBSITE USING ML"** under the guidance of **Mr. Umashanker Sharma**.

This project work has not been submitted anywhere for any degree.

Mr. Umashanker Sharma
Assistant Professor, CSE-AIML

Dr. Jay Shankar Prasad
HoD-AIML

Department of CSE (AI & ML)

Session 2024-2025

Project Completion Certificate

Date: 04/05/2025

This is to certify that **Mr. Anshuman Soni** bearing Roll No.2101321530007, student of 4th year CSE-AIML has completed project program (KCS-851) with the Department of CSE-AIML from 26-Feb-25 to 30-May-25.

He worked on the Project Titled **"ANONYMITY AND CONFIDENTIALITY IN WEBSITE USING ML"** under the guidance of **Mr. Umashanker Sharma**.

This project work has not been submitted anywhere for any degree.

Mr. Umashanker Sharma
Assistant Professor, CSE-AIML

Dr. Jay Shankar Prasad
HoD-AIML

Department of CSE (AI & ML)

Session 2024-2025

Project Completion Certificate

Date: 04/05/2025

This is to certify that **Mr. Anuj Mishra** bearing Roll No.2101321530008, student of 4th year CSE-AIML has completed project program (KCS-851) with the Department of CSE-AIML from 26-Feb-25 to 30-May-25.

He worked on the Project Titled **"ANONYMITY AND CONFIDENTIALITY IN WEBSITE USING ML"** under the guidance of **Mr. Umashanker Sharma**.

This project work has not been submitted anywhere for any degree.

Mr. Umashanker Sharma
Assistant Professor, CSE-AIML

Dr. Jay Shankar Prasad
HoD-AIML

Department of CSE (AI & ML)

Session 2024-2025

Project Completion Certificate

Date: 04/05/2025

This is to certify that **Mr. Shivam Singh** bearing Roll No.2101321530045, student of 4th year CSE-AIML has completed project program (KCS-851) with the Department of CSE-AIML from 26-Feb-25 to 30-May-25.

He worked on the Project Titled **"ANONYMITY AND CONFIDENTIALITY IN WEBSITE USING ML"** under the guidance of **Mr. Umashanker Sharma**.

This project work has not been submitted anywhere for any degree.

Mr. Umashanker Sharma
Assistant Professor, CSE-AIML

Dr. Jay Shankar Prasad
HoD-AIML

ACKNOWLEDGEMENT

We would like to sincerely thank **Mr. UmaShanker Sharma**, our project coordinator, and all of the professors for their counsel, inspiration, and unwavering support over the course of our project work. Without their assistance and insightful recommendations, our task would not have been feasible. We are deeply grateful to our esteemed Department Head, CSE-AIML **Dr. Jay Shankar Prasad**, for his counsel and assistance when needed.

We are also appreciative of **Dr. Dheeraj Gupta**, our director, for providing the resources we needed to complete our project job effectively.

We would like to express our gratitude to all of our friends for their support and helpful advice throughout this effort. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for everything they have given to us.

ABSTRACT

Phishing attacks are among the simplest yet most effective ways for cybercriminals to obtain sensitive information from unsuspecting users. These attacks target critical data such as usernames, passwords, and financial information, posing serious threats to individuals and organizations alike.

In response to this growing threat, cybersecurity experts are leveraging advanced detection techniques to identify phishing websites. This project harnesses machine learning technology to analyze features of legitimate and malicious URLs, offering a robust solution.

Three algorithms—Decision Tree, Random Forest, and Support Vector Machine—are implemented to evaluate performance. The primary goal is to identify the most reliable model by comparing metrics such as accuracy, false positive, and false negative rates, ensuring an efficient approach to combat phishing threats.

INDEX

TABLE OF CONTENT

CHAPTER	TITLE	PAGE
	CERTIFICATE	**I**
	ACKNOWLEDGEMENT	**IV**
	ABSTRACT	**V**
	LIST OF FIGURE	**VI**
	LIST OF TABLE	**VII**
1	**INTRODUCTION**	**8-11**
	1.1 INTRODUCTION	**08**
	1.2 PROBLEM STATEMENT	**09**
	1.3 IDENTIFICATION AND NEED	**09**
	1.4 OBJECTIVE	**10**
	1.5 UNIQUENESS OF THE INNOVIATION	**10**
	1.6 APPLICATIONS OF PROJECT	**11**
	1.6.1 POTENTIAL AREAS OF APPLICATION IN INDUSTRY/MARKET IN BRIEF	**11**
	1.6.2 MARKET POTENTIAL OF IDEA/INNOVIATION	**11**
2	**LITERATURE SURVEY**	**12-15**
	2.1 REVIEW OF LITERATURE	**12-15**
3	**PROBLEM FORMULATION AND PROPOSED WORK**	**16-19**
	3.1 PROBLEM STATEMENT	**16**
	3.2 PROPOSED SYSTEM	**16-17**
	3.3 ADVANTAGE OF PROPOSED SYSTEM	**18**
	3.4 LIMITATIONS	**19**
4	**FEASIBILITY STUDY**	**20**
	4.1 TECHNICAL FEASIBILITY	**20**
	4.2 ECONOMIC FEASIBILITY	**20**
5	**METHODOLOGY**	**21-30**
	5.1 METHODOLOGY	**21-30**
	5.1.2. DATA COLLECTION	**21-22**
	5.1.3. FEATURE EXTRACTION	**22-25**
	5.1.4. WORKING OF PHISH TANK	**25**
	5.1.5. TECHNOLOGIES USED	**26-27**
	5.1.6. PROCEDURE	**28-29**
	5.1.7. IMPLEMENTATION	**29**
	5.1.8. TESTING	**30**
6	**CODING**	**31-100**
	6.1. FRONT END PROGRAM	**31-81**
	6.2. BACK END PROGRAM	**82-100**
7	**RESULT AND DISCUSSION**	**101-103**

CHAPTER		PAGE
	7.1. Results	**101-102**
	7.2. DISCUSSION	**102-103**
8	**SCREENSHOTS**	**104-107**
		108-109
9	**FUTURE SCOPE AND CONCLUSION**	**108**
	9.1. CONCLUSION	**109**
	9.2. FUTURE SCOPE	
	REFERENCES	**100-111**

LIST OF FIGURES

Fig.3.2.1: **Model analysis of proposed system:** Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning **17**

Fig.5.1.1.i. phishing_urls.csv **22**

Fig.5.1.1.ii. legitimate_urls.csv **22**

Fig.5.1.4.i.. Flowchart of the Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning **27**

Fig.8.1. Output and link to open local webapp using Flask **104**

Fig.8.2. Youtube URL as legitimate website **104**

Fig.8.3. Facebook URL as legitimate website **105**

Fig.8.4. OnlineFix(3rd party gaming website) URL as Phishing website **105**

Fig.8.5. Irs taxrefund fake website URL as Phishing website **106**

Fig.8.6. Icici bank fake website URL as Phishing website **106**

Fig.8.6. Icici bank fake website URL as Phishing website **107**

Fig.8.8. McAfee web advisor's warning **107**

LIST OF TABLE

1. Table. 7.1.1.i:- Classifier's Performance **102**

Chapter 1

Introduction

1.1 Introduction

Phishing websites are carefully crafted online traps designed to steal sensitive information from unsuspecting users. They often mimic legitimate sites, luring victims to divulge login credentials, financial data, or other valuable personal details.

Nowadays Phishing has become a main area of concern for security researchers because it is not difficult to create a fake website which looks so close to a legitimate website. Experts can identify fake websites but not all the users can identify the fake website, and such users become the victim of phishing attack.

Main aim of the attacker is to steal banks account credentials. In United States businesses, there is a loss of US $2 billion per year because their clients become victim to phishing. In the 3rd Microsoft Computing Safer Index Report released in February 2014, it was estimated that the annual worldwide impact of phishing could be as high as $5 billion. Phishing attacks are becoming successful because of lack of user awareness. Since phishing attacks exploit the weaknesses found in users, it is difficult to mitigate them, but it is important to enhance phishing detection techniques.

The general method to detect phishing websites by updating blacklisted URLs, Internet Protocol (IP) to the antivirus database which is also known as “blacklist" method. To evade blacklists attackers uses creative techniques to fool users by modifying the URL to appear legitimate via obfuscation and many other simple techniques including fast flux, in which proxies are automatically generated to host the webpage; algorithmic generation of new URLs; etc.

A major drawback of this method is that it cannot detect zero-hour phishing attack. Heuristic based detection which includes characteristics that are found to exist in phishing attacks and can detect zero-hour phishing attack, but the characteristics are not guaranteed to always exist in such attacks and false positive rate in detection is very high.

To overcome the drawbacks of blacklist and heuristics-based method, many security researchers now focused on machine learning techniques. Machine learning technology consists of many algorithms

which require past data to decide or predict future data. Using this technique, algorithms will analyze various blacklisted and legitimate URLs and their features to accurately detect phishing websites including zero- hour phishing websites.

1.2 Problem Statement

Phishing websites have become a significant threat in the digital landscape, leading to financial losses and identity theft for individuals and organizations. These fraudulent websites often mimic legitimate ones, tricking users into disclosing sensitive information such as passwords, credit card numbers, and personal details. Traditional methods for detecting phishing sites, such as blacklisting, are limited in scalability and fail to detect new or dynamically generated phishing URLs in real-time.

The goal of this project is to develop a machine learning-based system to detect phishing websites by analysing features such as URL structure, domain age, security certificates, and website content. By leveraging supervised learning techniques, the system should be able to classify websites as phishing or legitimate with high accuracy, based on a set of labelled data.

1.3 Identification of Need

- With the rapid increase in internet users, phishing attacks have become one of the most prevalent cyber threats, tricking users into revealing sensitive information like login credentials, financial data, and personal details.
- Existing solutions such as browser filters, blacklists, and manual reporting are either reactive, outdated quickly, or insufficient against sophisticated and evolving phishing techniques.
- There is a need for a lightweight, automated, and real-time phishing detection system that is capable of accurately identifying phishing websites using machine learning, ensuring user protection without requiring technical knowledge or manual intervention.
- Phishing victims, especially those unfamiliar with cyber safety practices (e.g., elderly users, students, small business owners), require a proactive and easy-to-use defense mechanism to safeguard their online presence.

1.4 Objective

- Collect and preprocess data, including features from both phishing and legitimate websites.
- Apply feature engineering to identify key characteristics distinguishing phishing websites.
- Train multiple machine learning models and compare their performance.
- Develop an efficient and scalable solution that can identify phishing websites in real-time.
- Evaluate the model's accuracy, precision, recall, and F1-score using appropriate test datasets.
- The successful implementation of this project will result in a robust and efficient system capable of helping users and organizations safeguard against phishing attacks.

1.5 Uniqueness of the innovation

- The system leverages machine learning algorithms (Decision Tree, Random Forest, Support Vector Machine) to analyze URL-based features, enabling detection of phishing websites without relying on outdated blacklists or third-party databases.
- It operates in real-time and does not require access to the full website content, which makes it lightweight and faster than traditional phishing filters.
- The feature set used for classification is carefully selected to ensure high accuracy with low computational overhead, making the system deployable on both web and local environments.
- Unlike traditional tools, PhishShield offers comparative algorithm analysis, identifying the most effective model for phishing detection through accuracy and false positive/negative metrics.
- The project emphasizes ease of integration and scalability, allowing future expansion into browser extensions or enterprise-level security frameworks.

1.6 Applications of Project

1.6.1 Potential areas of application in industry/market in brief

The machine learning–based phishing detection system **PhishShield** provides various applications in the domain of cybersecurity, which are listed as follows:

- Offers a real-time, automated solution for detecting phishing websites, reducing user dependence on traditional, less adaptive blacklist-based detection methods.
- Helps secure online platforms such as banking portals, e-commerce websites, and educational institutions by preventing credential theft and data breaches.
- Can be integrated into corporate email and browsing systems to safeguard employee communications and reduce organizational risk from phishing campaigns.
- Provides end users, especially non-technical individuals, with a lightweight, user-friendly tool for protection against phishing attacks, increasing public awareness and safety.
- PhishShield's algorithm comparison module provides insights into the most efficient machine learning models for phishing detection, allowing security experts to fine-tune defense mechanisms.
- The platform can be extended to browser extensions, mobile applications, and even security awareness training tools, contributing to a layered defense approach against cyber threats.

1.6.2 Market potential of idea/innovation

- The proposed system can be adopted by cybersecurity solution providers as part of their intelligent threat detection offerings for organizations and individuals.
- PhishShield can be marketed to educational institutions, IT training centers, and government cyber awareness programs for simulation-based phishing detection training.
- The adaptability of the platform makes it suitable for deployment in small businesses and startups with limited budgets but high exposure to phishing threats.
- By enabling integration with browsers and mobile apps, PhishShield can enter the mass consumer market, offering affordable and accessible phishing protection to the general public.

Chapter 2

Literature Survey

2.1 Review of Literature

[1] Marwa Abd Al Hussein Qasim, Dr. Nahla Abbas Flayh: Using six criteria based on URL parameters such as the subdomain, principal domain, Page rank, Alexa rank, path domain, and Alexa reputation, this article suggests a novel method for identifying phishing websites. The method focuses on evaluating how closely a phishing site's URL resembles the URL of a reliable website and also takes into account the site's ranking as a crucial component in determining its validity. The approach was tested using data from PhishTank and DMOZ, and the authors showed that it could identify over 97% of phishing sites.

[2] Irfan Siddavatam, Rishikesh Mahajan: Introduced the use of decision trees , SVM, Random Forest to classify phishing websites based on features like URL length, domain age, presence of special characters, and security certificate information. Their approach demonstrated improved detection rates compared to traditional methods but faced challenges in identifying phishing sites.

[3] S. Arvind Anwekar, V. Agrawal : In this study, the authors focused on extracting features from URLs, in addition to other features such as the age of the SSL certificate and the universal resource locator of the anchor, IFRAME, and website rank. They collected URLs of phishing websites from PhishTank and URLs of benign websites from the Alexa website. Using a combination of the random forest (RF), decision tree (DT), and support vector machine (SVM), contributed to improving the detection mechanism for phishing websites and achieved a high noticeable detection accuracy of 97.14%, with a low rate of false positives at 3.14%. The results also showed that the classifier's performance improves with more training data.

[4] N. Choudhary b, K. Jain, S. Jain : This study emphasizes the significance of only using attributes from the URL. Both the Kaggle and Phishtank websites make it easy to get the dataset used in this study. The researchers used a hybrid approach that com-bined Principal Component Analysis (PCA) with Support Vector Machine (SVM) and Random Forest algorithms to reduce the dataset's dimensionality while keeping all im-portant data, and it produced a higher accuracy rate of 96.8% compared to other tech-niques investigated.

[5] A. Lakshmanarao, P. Surya, M Bala Krishna: This thesis collected a dataset of phishing websites from the UCI repository and used various Machine learning techniques, including decision trees, AdaBoost, support vector machines (SVM), and random forests, to analyze selected features (such as web traffic, port,

URL length, IP address, and URL_of_Anchor). The most effective model for detecting phishing web-sites was chosen, and two priority-based algorithms (PA1 and PA2) were proposed. The team utilized a new fusion classifier in conjunction with these algorithms and at-tained an accuracy rate of 97%. when compared to previous works in phishing website detection

.

[6] L. Tang, Q. Mahmoud: The proposed approach in the current study uses URLs collected from a variety of platforms, including Kaggle, Phish Storm, Phish Tank, and ISCX-UR, to identify phishing websites. The researchers made a big contribution since they created a browser plug-in that can quickly recognize phishing risks and offer warn-ings. Various datasets and machine learning techniques were investigated, and the pro-posed RNN-GRU model outperformed SVM, Random Forest (RF), and Logistic Re-gression with a maximum accuracy rate of 99.18%. On the other hand, the suggested method is not always accurate in identifying if short links are phishing risks.

[7] A. Kulkarni & L. Brown: A machine learning system was created to categorize websites based on URLs from the University of California, Irvine Machine Learning Repository. Four classifiers were used: SVM, decision tree, Naive Bayesian, and neural network. The outcome of experiments utilizing the model developed with the support of a training set of data demonstrates that the classifiers were able to successfully differentiate authentic websites from fake ones with an accuracy rate of over 90%. Limitations include a small dataset and all features being discrete, which may not be suitable for some classifiers.

[8] Tyagi; J. Shad; S. Sharma; S. Gaur Gagandeep Kaur: The research taken into account focuses on the use of various machine learning algorithms to identify if a web-site is legitimate or a phishing site based on a URL. This study's most important con-tribution is the creation of the Generalized Linear Model (GLM), a brand-new model. This model combines the results of two various methods. With a 98.4% accuracy rate, the Random Forest and GLM combination produced the best results for detecting phishing websites.

[9] M. Karabatak and T. Mustafa: The objective of this research is to assess the effectiveness of classification algorithms on a condensed dataset of phishing websites obtained from the UCI Machine Learning Repository. The paper investigates how data mining and feature selection algorithms affect reduced datasets through experiments and analysis, finally selecting the methods that perform the best in terms of classifica-tion. According to the results, some classification strategies improve performance while others have the opposite impact. Ineffective classifiers for condensed phishing datasets included Lazy, BayesNet, SGD Multilayer Perceptron, PART, JRip, J48, RandomTree, and RandomForest. However, it was discovered that KStar,

LMT, ID3, and R.F.Clas-sifier were efficient. Lazy produced the highest classification accuracy rate of 97.58% on the compressed 27-feature dataset, whereas KStar performed at its best on the same dataset.

[10] X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng: A phishing detection model that applies Bagging, AdaBoost, SMO, and Random Forest algorithms to learn and test phishing detection strategies is offered as a contribution to this work. The model is based on features from URLs and extracts multi-level statistical characteristics, seman-tic features of word embedding, and semantic features from Chinese web content. Legal URLs from DirectIndustry online instructions and phishing data from the Anti-Phishing Alliance of China (APAC) are included in the dataset used to test the algorithm. The study's findings suggest that a fusion model that primarily employed semantic data to identify phishing sites with high detection efficiency had the best performance, leading to a new contribution with an F-measure of 0.99%. Keep in mind that this approach is specific to Chinese websites and is language-dependent.

[11] W. Fadheel, M. Abusharkh, and I. Abdel-Qader : The present study utilized datasets from the UCI machine learning repository, including Domain, HTML, Address Bar, and URLs, the main contribution was conducting a comparative analysis of the impact of feature selection on detecting phishing websites. The KMO test was applied in the study to evaluate the dataset using (LR) and (SVM) classification algorithms. The test was conducted based on a correlation matrix to analyze the performance. Re-sults showed that LR with the KMO test achieved an accuracy of 91.68%, while SVM with the KMO test yielded an accuracy of 93.59%.

[12] A. Ahmed and N. A. Abdullah: The research team developed a software program known as Phish Checker, which is designed to distinguish between legitimate and phishing websites. The proposed approach focuses on identifying phishing attacks by analyzing the URLs and domain names of suspected phishing websites to determine their authenticity. Data was collected from the Yahoo and PhishTank directories and the results indicate that PhishChecker has an accuracy rate of 96% for identifying phish-ing websites. However, it should be noted that this method is based on heuristics and its effectiveness is reliant on the availability of certain discriminative elements that aid in identifying the type of website. Additionally, the study only examines the validity of URLs in determining website authenticity.

[13] Ankit Kumar Jain & B. B. Gupta: The proposed strategy utilizes an Innovative methodology for defending counteract phishing attempts by incorporating a URL and DNS matching module with a white list of trusted websites that are automatically up-dated based on each user's browsing history. This method offers quick retrieval speeds, high rates of detection, and alerts users to not disclose personal information when at-

tempting to access a website, not on the white list. Additionally, hyperlink properties are utilized to verify the validity of a website by retrieving hyperlinks from the source code and applying them to the phishing detection method. The performance of this strategy was evaluated using data from reputable sources such as Stuffgate, Alexa, and PhishTank and achieved an accuracy rate of 89.38 %.

[14] M. Aydin and N. Baykal: Throughout this experiment, phishing websites were detected using subset-based feature selection methods based on URL attributes. A da-taset comprising both legitimate and phishing URLs was obtained from Google and PhishTank, and multiple features were retrieved from URLs. The usefulness of two classification algorithms—Naive Bayes and Sequential Minimal Optimization (SMO)—for identifying phishing websites was investigated in this study. The results showed that SMO performed better than Naive Bayes for phishing detection, with an accuracy rate of 95.39%. The SMO algorithm also had another benefit in that it made use of more chosen features overall. The accuracy rate of the Naive Bayes method, in contrast, was 88.17% while using the same quantity of chosen features.

Chapter 3

Problem Formulation and Proposed Work

3.1 Problem Statement

Phishing websites have become a significant threat in the digital landscape, leading to financial losses and identity theft for individuals and organizations. These fraudulent websites often mimic legitimate ones, tricking users into disclosing sensitive information such as passwords, credit card numbers, and personal details. Traditional methods for detecting phishing sites, such as blacklisting, are limited in scalability and fail to detect new or dynamically generated phishing URLs in real-time.

The goal of this project is to develop a machine learning-based system to detect phishing websites by analysing features such as URL structure, domain age, security certificates, and website content. By leveraging supervised learning techniques, the system should be able to classify websites as phishing or legitimate with high accuracy, based on a set of labelled data.

3.2 Proposed System

To address the growing threat of phishing attacks, we propose **PhishShield** – a machine learning–based detection system that analyzes URL features to identify and block phishing websites in real time.

• **PhishShield** is an intelligent cybersecurity tool that has the potential to significantly reduce phishing incidents by proactively identifying malicious URLs before they can cause harm.
• The proposed system uses machine learning algorithms such as Decision Tree, Random Forest, and Support Vector Machine to classify URLs based on specific lexical and structural features. This automated detection method offers faster and more scalable protection compared to traditional blacklist-based solutions.
• PhishShield extracts key features from input URLs—such as length, use of special characters, presence of suspicious keywords, and domain information—and feeds them into the selected machine learning model to determine whether a URL is legitimate or phishing.
• The core components of PhishShield include a feature extraction module, a trained machine learning classifier, and a prediction interface that alerts users when a malicious site is detected. These components are built using Python, with support from libraries like Scikit-learn and Pandas for model development and evaluation.

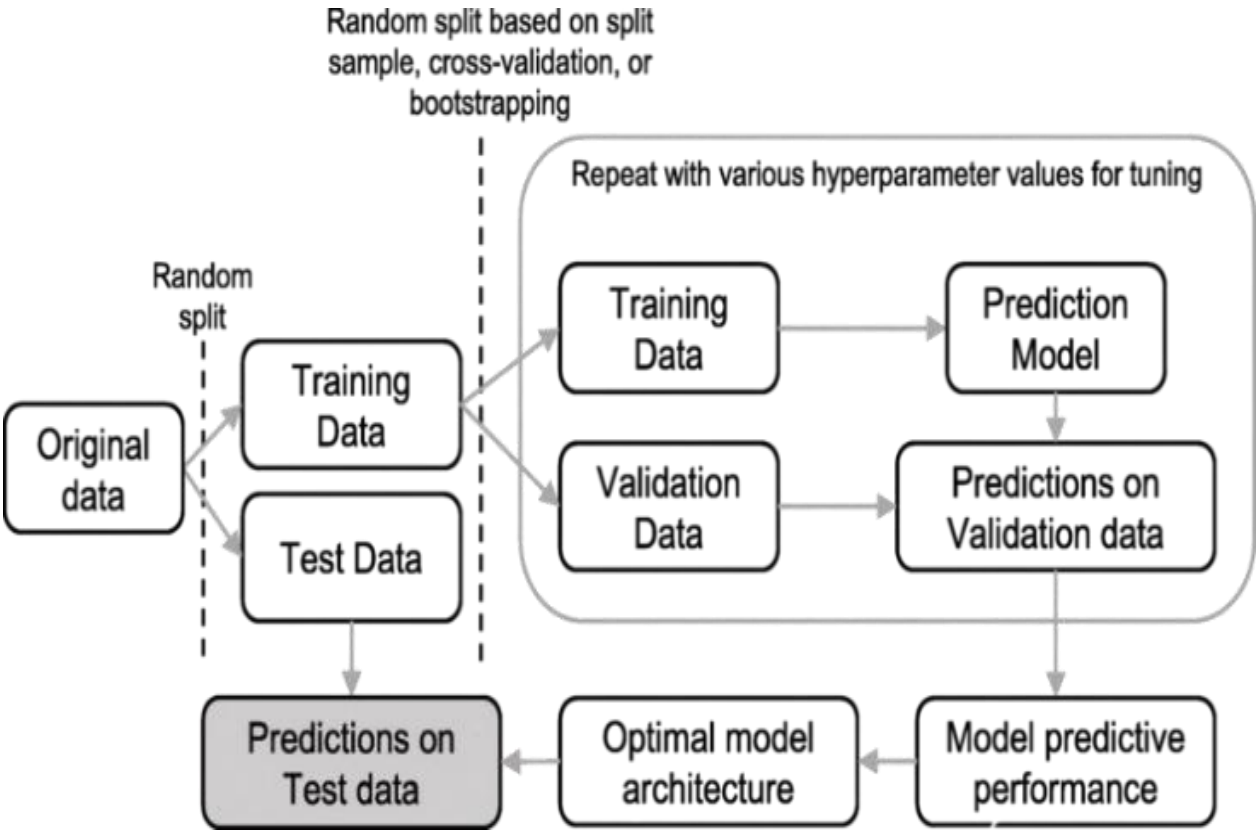


Fig.3.2.1: **Model analysis of proposed system:** Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning

3.3 ADVANTAGE OF PROPOSED SYSTEM

Advantages of the Proposed System:

1. **Real-time Detection:**

   The system can analyse websites in real-time, providing instant feedback on whether a website is legitimate or a phishing attempt.

2. **High Accuracy:**

   Machine learning models can achieve high accuracy by learning from vast datasets of phishing and legitimate websites, minimizing false positives and negatives.

3. **Adaptive to New Threats:**

   Unlike traditional blacklist-based systems, the tool can detect newly created or dynamically generated phishing websites by recognizing patterns and features associated with phishing attempts.

4. **Scalability:**

   The tool can be easily scaled to analyse many websites simultaneously, making it suitable for both individual users and organizations.

5. **Low Maintenance:**

   Once trained, the system can operate with minimal human intervention, only requiring periodic retraining with new data to stay updated with the latest phishing trends.

6. **Platform Independence:**

   Can be integrated into various platforms such as browsers, antivirus software, or even mobile applications, providing flexible usage across different systems.

7. **Cost-Effective:**

   Reduces the need for expensive manual oversight and cybersecurity teams dedicated to phishing detection, lowering the overall cost for businesses.

8. **Reduced Human Error:**

   Automated detection significantly reduces human errors, which can occur during manual inspection of potentially suspicious websites.

3.4 Limitations

1. The system may generate false positives or false negatives depending on URL similarities with legitimate domains.
2. It requires labeled datasets for model training, which may need regular updates to remain effective against evolving threats.
3. Does not analyze full webpage content, which may limit detection accuracy in highly obfuscated phishing attempts.

Performance may vary depending on the quality and size of the dataset used for training.

Chapter 4

FEASIBILITY STUDY

4.1. Technical Feasibility

1. The project is technically feasible as it uses well-established machine learning algorithms (Decision Tree, Random Forest, and SVM) and is built using Python, a widely used programming language for data science and cybersecurity applications.
2. It does not require any specialized hardware, as the system runs efficiently on standard computing environments.
3. PhishShield leverages URL-based feature extraction, which is simple to implement and does not involve complex content parsing or web scraping.
4. Available libraries such as Scikit-learn, Pandas, and NumPy support the development, testing, and optimization of the model, ensuring ease of implementation and maintenance.

4.2 ECONOMIC FEASIBILITY

1. The development cost of the proposed system is low, as it relies on open-source tools and free datasets for training and testing.
2. expensive hardware or third-party cybersecurity subscriptions are required, making it cost-effective for individual users and small organizations.
3. The system can be deployed as a browser extension or desktop application without significant licensing costs.
4. Maintenance costs are minimal, and updates can be provided as part of regular software patches, ensuring long-term affordability and scalability.

Chapter 5

Methodology

5.1 Methodology

5.1.1. DATA COLLECTION

a. Data Collection:

Phishing Website Data: Gather datasets from publicly available sources such as PhishTank, UCI repository, or Kaggle, which contain records of both phishing and legitimate websites.

Legitimate Website Data: Collect data from trustworthy sources for a balanced comparison with phishing websites.

b. Data sets:

- The set of phishing URLs are collected from opensource service called **PhishTank**. This service

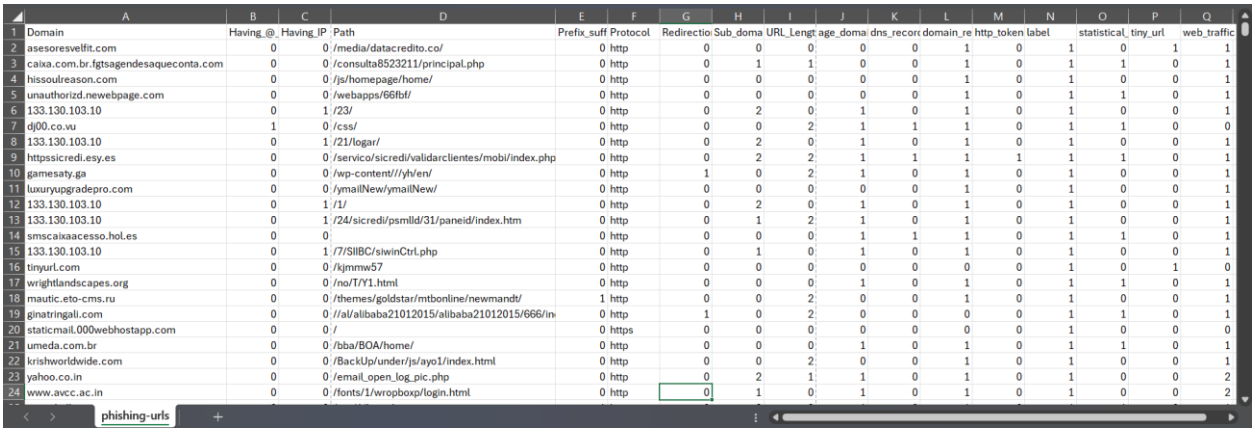
  provide a set of phishing URLs in multiple formats like csv, json etc. that gets updated hourly. To

  download the data: https://www.phishtank.com/developer_info.php. From this dataset, 5000 random phishing URLs are collected to train the ML models.
- The legitimate URLs are obatined from the open datasets of the University of New Brunswick, https://www.unb.ca/cic/datasets/url-2016.html. This dataset has a collection of benign,

  spam, phishing, malware & defacement URLs. Out of all these types, the benign url dataset is considered for this project. From this dataset, 5000 random legitimate URLs are collected to train the ML models.

c. Data Sets Sample:

Sample of Phishing websites csv

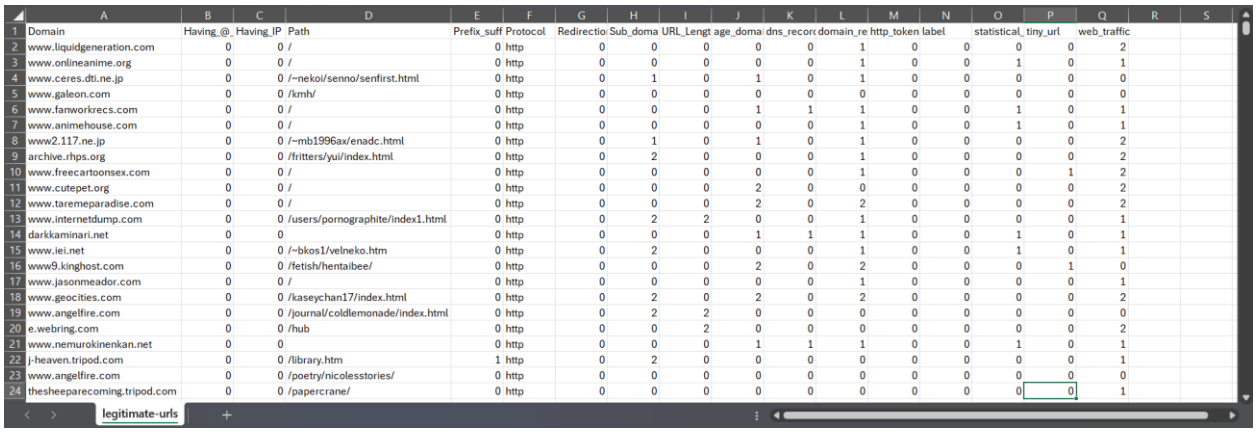


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Domain	Having_@	Having_IP	Path	Prefix_suff	Protocol	Redirectio	Sub_doma	URL_Lengt	age_doma	dns_recor	domain_re	http_token	label	statistical_	tiny_url	web_traffic
2	asesoresvelfit.com	0	0	/media/datacredito.co/	0	http	0	0	0	0	0	1	0	1	0	1	1
3	caixa.com.br.fgtsagendesaqueconta.com	0	0	/consulta8523211/principal.php	0	http	0	1	1	0	0	1	0	1	1	0	1
4	hissoulreason.com	0	0	/js/homepage/home/	0	http	0	0	0	0	0	1	0	1	0	0	1
5	unauthorizd.newebpage.com	0	0	/webapps/66fbf/	0	http	0	0	0	0	0	1	0	1	1	0	1
6	133.130.103.10	0	1	/23/	0	http	0	2	0	1	0	1	0	1	0	0	1
7	dj00.co.vu	1	0	/css/	0	http	0	0	2	1	1	1	0	1	1	0	0
8	133.130.103.10	0	1	/21/logar/	0	http	0	2	0	1	0	1	0	1	0	0	1
9	httpssicredi.esy.es	0	0	/servico/sicredi/validarclientes/mobi/index.php	0	http	0	2	2	1	1	1	1	1	1	0	1
10	gamesaty.ga	0	0	/wp-content///yh/en/	0	http	1	0	2	1	0	1	0	1	0	0	1
11	luxuryupgradepro.com	0	0	/ymailNew/ymailNew/	0	http	0	0	0	0	0	1	0	1	0	0	1
12	133.130.103.10	0	1	/1/	0	http	0	2	0	1	0	1	0	1	0	0	1
13	133.130.103.10	0	1	/24/sicredi/psmlld/31/paneid/index.htm	0	http	0	1	2	1	0	1	0	1	0	0	1
14	smscaixaacesso.hol.es	0	0		0	http	0	0	0	1	1	1	0	1	1	0	1
15	133.130.103.10	0	1	/7/SIIBC/siwinCtrl.php	0	http	0	1	0	1	0	1	0	1	0	0	1
16	tinyurl.com	0	0	/kjmmw57	0	http	0	0	0	0	0	0	0	1	0	1	0
17	wrightlandscapes.org	0	0	/no/T/Y1.html	0	http	0	0	0	1	0	1	0	1	1	0	1
18	mautic.eto-cms.ru	0	0	/themes/goldstar/mtbonline/newmandt/	1	http	0	0	2	0	0	1	0	1	0	0	1
19	ginatringali.com	0	0	//al/alibaba21012015/alibaba21012015/666/in	0	http	1	0	2	0	0	0	0	1	1	0	1
20	staticmail.000webhostapp.com	0	0	/	0	https	0	0	0	0	0	0	0	1	0	0	0
21	umeda.com.br	0	0	/bba/BOA/home/	0	http	0	0	0	1	0	1	0	1	1	0	1
22	krishworldwide.com	0	0	/BackUp/under/js/ayo1/index.html	0	http	0	0	2	0	0	1	0	1	0	0	1
23	yahoo.co.in	0	0	/email_open_log_pic.php	0	http	0	2	1	1	0	1	0	1	0	0	2
24	www.avcc.ac.in	0	0	/fonts/1/wropboxp/login.html	0	http	0	1	0	1	0	1	0	1	0	0	2

phishing-urls

Fig.5.1.1.i. phishing_urls.csv

Sample of Legitimate websites csv



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Domain	Having_@	Having_IP	Path	Prefix_suff	Protocol	Redirectio	Sub_doma	URL_Lengt	age_doma	dns_recor	domain_re	http_token	label	statistical_	tiny_url	web_traffic		
2	www.liquidgeneration.com	0	0	/	0	http	0	0	0	0	0	1	0	0	0	0	2		
3	www.onlineanime.org	0	0	/	0	http	0	0	0	0	0	1	0	0	1	0	1		
4	www.ceres.dti.ne.jp	0	0	/~nekoi/senno/senfirst.html	0	http	0	1	0	1	0	1	0	0	0	0	0		
5	www.galeon.com	0	0	/kmh/	0	http	0	0	0	0	0	0	0	0	0	0	0		
6	www.fanworkrecs.com	0	0	/	0	http	0	0	0	1	1	1	0	0	1	0	1		
7	www.animehouse.com	0	0	/	0	http	0	0	0	0	0	1	0	0	1	0	1		
8	www2.117.ne.jp	0	0	/~mb1996ax/enadc.html	0	http	0	1	0	1	0	1	0	0	0	0	2		
9	archive.rhps.org	0	0	/fritters/yui/index.html	0	http	0	2	0	0	0	1	0	0	0	0	2		
10	www.freecartoonsex.com	0	0	/	0	http	0	0	0	0	0	1	0	0	0	1	2		
11	www.cutepet.org	0	0	/	0	http	0	0	0	2	0	0	0	0	0	0	2		
12	www.taremeparadise.com	0	0	/	0	http	0	0	0	2	0	2	0	0	0	0	2		
13	www.internetdump.com	0	0	/users/pornographite/index1.html	0	http	0	2	2	0	0	1	0	0	0	0	1		
14	darkkaminari.net	0	0		0	http	0	0	0	1	1	1	0	0	1	0	1		
15	www.iei.net	0	0	/~bkos1/velneko.htm	0	http	0	2	0	0	0	1	0	0	1	0	1		
16	www9.kinghost.com	0	0	/fetish/hentaibee/	0	http	0	0	0	2	0	2	0	0	0	1	0		
17	www.jasonmeador.com	0	0	/	0	http	0	0	0	0	0	1	0	0	0	0	1		
18	www.geocities.com	0	0	/kaseychan17/index.html	0	http	0	2	0	2	0	2	0	0	0	0	2		
19	www.angelfire.com	0	0	/journal/coldlemonade/index.html	0	http	0	2	2	0	0	0	0	0	0	0	0		
20	e.webring.com	0	0	/hub	0	http	0	0	2	0	0	0	0	0	0	0	2		
21	www.nemurokinenkan.net	0	0		0	http	0	0	0	1	1	1	0	0	1	0	1		
22	j-heaven.tripod.com	0	0	/library.htm	1	http	0	2	0	0	0	0	0	0	0	0	1		
23	www.angelfire.com	0	0	/poetry/nicolesstories/	0	http	0	0	0	0	0	0	0	0	0	0	0		
24	thesheeparecoming.tripod.com	0	0	/papercrane/	0	http	0	0	0	0	0	0	0	0	0	0	1		

legitimate-urls

Fig.5.1.1.ii. legitimate_urls.csv

5.1.2. Feature Extraction

Feature Extraction consists of:

Extract significant features from website URLs and webpage content that differentiate phishing websites from legitimate ones.

Key features to consider:

- URL-based features: Length of URL, presence of special characters, use of IP address instead of a domain name, presence of suspicious keywords.
- Domain-based features: Domain age, domain name length, registration information, presence of SSL certificates.
- Page-based features: Website content, use of HTTPS, form handling, third-party resources.

The below mentioned category of features are extracted from the URL data:

1. Address Bar based Features

In this category 9 features are extracted.

2. Domain based Features

In this category 4 features are extracted.

3. HTML & Javascript based Features

In this category 4 features are extracted.

So, all together 17 features are extracted from the 10,000 URL dataset.

Feature Extraction for Phishing Website Detection:

1. Using URL Shortening Services

- **Objective**: Identify phishing attempts using shortened URLs.
- **Method**: Detect URL shortening services like "bit.ly" or "tinyurl.com" through keyword matching. These services obscure the true destination, often used by attackers.

2. Existence of HTTPS Token

- **Objective**: Avoid misleading security indicators in the domain.
- **Method**: Flag URLs containing “HTTPS” in places other than the protocol (e.g., "httpssecure.com").

3. Abnormal URL

- **Objective**: Validate the legitimacy of domains.
- **Method**: Cross-check domains against WHOIS databases to verify registration details. Abnormal or unregistered domains raise suspicion.

4. Google Index

- **Objective**: Confirm the URL is indexed by search engines.
- **Method**: Use Google search APIs to verify if the URL exists in Google's database. Non-indexed sites are more likely to be phishing.

5. Website Traffic

- **Objective**: Assess website popularity.
- **Method**: Analyze Alexa rank data. Legitimate sites usually have higher traffic, while phishing sites rank low or are absent.

6. Domain Registration Length

- **Objective**: Detect short-lived domains.
- **Method**: Phishing domains often register for under a year. WHOIS checks identify short registration durations.

7. Age of Domain

- **Objective**: Identify newly created domains.
- **Method**: WHOIS records reveal the domain's age. New domains are a common indicator of phishing attempts.

8. DNS Record

- **Objective**: Verify domain validity.
- **Method**: DNS queries ensure the existence of legitimate records. Missing or invalid records suggest phishing activity.

9. Statistical Report

- **Objective**: Leverage threat intelligence databases.
- **Method**: Match URLs and IPs against lists of known phishing entities. Regularly updated datasets enhance accuracy.

10. Long URLs

- **Objective**: Detect unusually long URLs.
- **Method**: URLs exceeding a standard character limit (e.g., >75 characters) are flagged as suspicious.

11. @ Symbol in URL

- **Objective**: Identify domain obfuscation techniques.
- **Method**: The "@" symbol redirects users to different pages and is typically used in phishing URLs.

12. Double Slashes (//) in Path

- **Objective**: Detect abnormal URL formatting.
- **Method**: URLs with "//" in unexpected places (other than "http://") are flagged as suspicious.

13. Subdomains

- **Objective**: Identify excessive subdomains.

- **Method**: Count the number of dots in the URL. Phishing URLs often use multiple subdomains to confuse users (e.g., "login.bank.com.fake.com").

14. IP Address Usage

- **Objective**: Detect direct IP addresses in URLs.
- **Method**: URLs using raw IPs instead of domain names are flagged, as legitimate sites typically avoid this practice.

5.1.3. Working of Phishtank creation of dataset for train phishing website models

1. Feature Extraction

Machine learning models use different types of features to distinguish between phishing and legitimate websites. These features fall into three categories:

A. URL-Based Features (Static Analysis)

Extracted from the URL string itself:

- **Length of URL** (Shorter URLs are often legitimate, extremely long URLs may be phishing)
- **Use of IP Address Instead of Domain** (http://192.168.1.1/login.php → Likely phishing)
- **Presence of "@" Symbol** (Redirects user, common in phishing)
- **Number of Dots (.) in URL** (Excessive subdomains like login.bank-secure.com.phish.com)
- **URL Shortening Service Used** (bit.ly, tinyurl, etc. → Often used in phishing)

B. Domain-Based Features (DNS & WHOIS Data)

Extracted from domain records:

- **Domain Age** (Newly registered domains are suspicious)
- **Domain Expiry** (Short expiration time is a phishing indicator)
- **Alexa Rank** (Legitimate websites usually have higher ranks)
- **HTTPS Presence** (Phishing sites often lack SSL certificates)

C. HTML & JavaScript-Based Features (Page Content Analysis)

Extracted from the website's source code:

- **Using IFrames** (Hides the real URL to deceive users)
- **Right-Click Disabled** (Prevents users from checking site security details)
- **Form Submits to Different Domain** (Login page sends credentials to an unknown website)

5.1.4. Technologies Used

Detecting phishing websites using machine learning algorithms typically involves using supervised learning techniques. Some common machine learning algorithms and methods used for phishing website detection include:

1. Logistic Regression: This algorithm is commonly used for binary classification tasks like phishing detection, where the output is either phishing or legitimate.
Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable.

2. Decision Trees: Decision trees can be effective for phishing detection as they can capture complex relationships between features. Ensemble methods like Random Forests and Gradient Boosting Machines (GBM) can also be used for improved performance.
A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical tree structure, which consists of a root node, branches, internal nodes, and leaf nodes.

3. Support Vector Machines (SVM): SVMs are good for binary classification tasks and can be trained to distinguish between phishing and legitimate websites based on various features.
Support vector machine is another powerful algorithm in machine learning technology. In support vector machine algorithm, each data item is plotted as a point in n-dimensional space and support vector machine algorithm constructs separating line for classification of two classes, this separating line is well known as hyperplane. The Support vector machine seeks for the closest points called as support vectors and once it finds the closest point it draws a line connecting to them. Support vector machine then construct separating line which bisects and perpendicular to the connecting line. To classify data perfectly the margin should be maximum. Here the margin is a distance between hyperplane and support vectors. In real scenario it is not possible to separate complex and nonlinear data, to solve this problem support vector machine uses kernel trick which transforms lower dimensional space to higher dimensional space.

4. K-Nearest Neighbors (KNN): KNN is a simple and intuitive algorithm that can be used for phishing detection by considering the similarity between features of websites.

The k-nearest neighbors (KNN) algorithm is a machine learning algorithm that uses proximity to classify or predict a data point's grouping. It is a supervised learning algorithm that works by comparing a data point to a set of data it was trained on.

5. Random Forest: Random Forest (RF) is a machine learning algorithm that combines the results of multiple decision trees to produce a single result.

Random forest algorithm is one of the most powerful algorithms in machine learning technology and it is based on the concept of decision tree algorithm. Random forest algorithm creates the forest with number of decision trees. High number of trees gives high detection accuracy. Creation of trees is based on bootstrap method. In bootstrap method features and samples of dataset are randomly selected with replacement to construct single tree.

Among randomly selected features, random forest algorithm will choose best splitter for the classification and like decision tree algorithm; Random Forest algorithm also uses Gini index and information gain methods to find the best splitter. This process will continue until random forest creates n number of trees. Each tree in forest predicts the target value and then algorithm will calculate the votes for each predicted target. Finally random forest algorithm considers high voted predicted target as a final prediction.

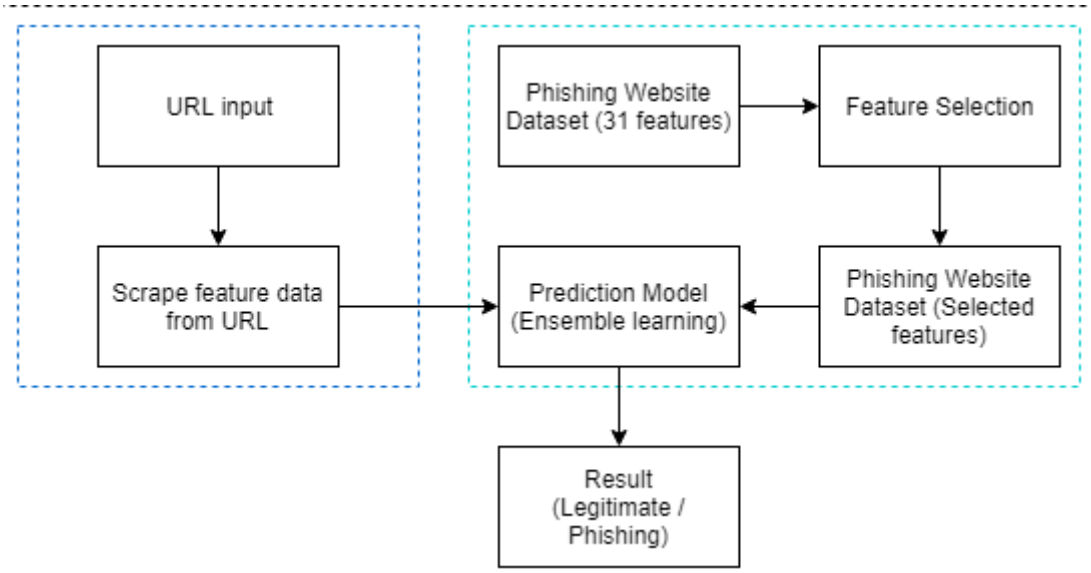


Fig.5.1.4.i.. Flowchart of the Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning

5.1.5. PROCEDURE

1. Data Collection:

- Collect URL data from different sources, including legitimate and phishing websites. Files like legitimate_urls.txt and 1000-phishing.txt provide the URLs used for training.

2. Feature Extraction (Class FeatureExtraction):

- **Protocol**: Extract whether the URL uses "http" or "https".
- **Domain**: Extract the domain name of the URL.
- **Path**: Extract the path from the URL.
- **Having IP**: Check if the domain contains an IP address.
- **URL Length**: Measure the length of the URL to classify it as suspicious or legitimate.
- **@ Symbol**: Check for the presence of "@" in the URL.
- **Redirection**: Detect if the URL contains a "//" after the protocol.
- **Subdomains**: Count the number of subdomains in the URL.
- **Shortening Services**: Detect URLs shortened using services like bit.ly, goo.gl.
- **Web Traffic**: Use Alexa rank to check the popularity of the domain.
- **Domain Registration Length**: Extract domain registration and compare the validity.
- **Age of Domain**: Calculate the age of the domain based on registration date.
- **DNS Records**: Check DNS information for additional classification.
- **Statistical Report**: Conduct a statistical analysis on the URL to detect anomalies.
- **HTTPS Token**: Check if the URL starts with "https://" for security validation.

3. Data Labeling:

- URLs are labeled as:
    - 0: Legitimate
    - 1: Phishing
    - 2: Suspicious

4. Model Training:

- Use RandomForestClassifier or DecisionTreeClassifier for the model to classify URLs as phishing or legitimate based on extracted features.

5. Web Interface:

- The app.py file likely serves the model using a web framework (e.g., Flask) and includes UI files like home.html, _navbar.html, and about.html for user interaction.

6. Model Saving and Loading:

- After training, save the model to a file like RandomForestModel.sav for future use.

7. Prediction:

- For new URLs, the system extracts features, processes them, and uses the trained model to classify the URL as phishing or legitimate.
-

5.1.6. Implementation

The script is designated to detect phishing websites by extracting specific features from URLs. The key steps involved:

1. Feature Extraction Class:
    - The class FeatureExtraction defines methods for various URL characteristics, such as:
        - Protocol (e.g., HTTP, HTTPS)
        - Domain: Checks if the domain is an IP address or a typical domain name.
        - URL Length: Longer URLs are often used in phishing sites.
        - Subdomains: More than three subdomains might indicate a phishing attempt.
        - Redirection and Symbols: Phishing sites often use tricks like @ symbols or redirect symbols (//).
        - Tiny URLs: Shortened URLs often lead to phishing sites.
2. WHOIS Lookup: The script uses the whois library to fetch domain registration details such as the domain's age and expiration date, which help assess if a website is legitimate. Short-lived or recently created domains are red flags.
3. Web Traffic Analysis: It checks the popularity of a site based on its Alexa ranking, as legitimate sites tend to have higher traffic.
4. CSV File Generation: After processing each URL from a file, the script stores the extracted features in a DataFrame and saves the data into a CSV file labeled as "phishing-urls.csv."
5. Labeling: The URLs are labeled based on the extracted features, helping in the classification of the website as legitimate, phishing, or suspicious.

This approach can be used to train machine learning models to automate phishing detection.

5.1.7. Testing

In the **Phishing Website Detection** project, testing the model involves the following steps:

1. **Data Splitting**: The dataset is divided into training and testing sets, typically using an 80-20 or 70-30 ratio. This ensures that the model is evaluated on unseen data.
2. **Model Evaluation**: After training the classifier (e.g., RandomForest or DecisionTree), the model is tested using the testing dataset. Performance is evaluated using metrics such as:
    - Accuracy
    - Precision
    - Recall
    - F1-Score
3. **Cross-validation**: To ensure robustness, cross-validation techniques like k-fold cross-validation may be applied to assess the model's performance across multiple subsets of the data.
4. **Confusion Matrix**: A confusion matrix helps visualize the classification results, showing true positives, false positives, true negatives, and false negatives.
5. **Performance Visualization**: The performance metrics, such as accuracy and F1-score, are plotted to assess how well the model is generalizing.

7. **Model Tuning**: Hyperparameters of the classifier can be adjusted (e.g., number of trees in Random Forest) to improve the model’s performance on the test set.

Chapter 6

Coding

6.1. FRONT END PROGRAM:

- **home.html**

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>PhishShield | Anonymity and Confidentiality in Website using Machine Learning</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap"
rel="stylesheet">

<style>

:root {

--primary: #4F46E5;

--primary-light: #6366F1;

--primary-dark: #4338CA;

--danger: #EF4444;

--success: #10B981;

--warning: #F59E0B;

--dark: #1F2937;

--light: #F9FAFB;
```

```
--gray: #6B7280;
--light-gray: #E5E7EB;
--white: #FFFFFF;
--shadow-sm: 0 1px 3px rgba(0, 0, 0, 0.1);
--shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
--shadow-md: 0 10px 15px rgba(0, 0, 0, 0.1);
--shadow-lg: 0 20px 25px rgba(0, 0, 0, 0.1);
--transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}
* {
margin: 0;
padding: 0;
box-sizing: border-box;
}
body {
font-family: 'Poppins', sans-serif;
line-height: 1.6;
color: var(--dark);
background-color: var(--light);
}
.container {
width: 100%;
max-width: 1200px;
```

```
margin: 0 auto;
padding: 0 1.5rem;
}
/* Header */
header {
background-color: var(--white);
box-shadow: var(--shadow-sm);
position: sticky;
top: 0;
z-index: 100;
}
.navbar {
display: flex;
justify-content: space-between;
align-items: center;
padding: 1.5rem 0;
}
.logo {
display: flex;
align-items: center;
gap: 0.75rem;
font-size: 1.5rem;
font-weight: 700;
```

```
text-decoration: none;
color: var(--primary-dark);
}
.logo-icon {
font-size: 1.75rem;
color: var(--primary);
}
.nav-links {
display: flex;
gap: 2rem;
}
.nav-link {
color: var(--gray);
text-decoration: none;
font-weight: 500;
transition: var(--transition);
position: relative;
}
.nav-link:hover {
color: var(--primary);
}
.nav-link.active {
color: var(--primary);
```

```
}
.nav-link.active::after {
content: '';
position: absolute;
bottom: -6px;
left: 0;
width: 100%;
height: 2px;
background-color: var(--primary);
}
.mobile-menu-btn {
display: none;
background: none;
border: none;
color: var(--dark);
font-size: 1.5rem;
cursor: pointer;
}
/* Hero Section */
.hero {
padding: 5rem 0;
background: linear-gradient(135deg, rgba(79, 70, 229, 0.03) 0%, rgba(255, 255, 255, 1) 100%);
position: relative;
```

```
overflow: hidden;
}
.hero-content {
display: flex;
flex-direction: column;
align-items: center;
text-align: center;
position: relative;
z-index: 1;
}
.hero-title {
font-size: 3.5rem;
font-weight: 700;
margin-bottom: 1.5rem;
line-height: 1.2;
background: linear-gradient(to right, var(--primary), var(--primary-light));
-webkit-background-clip: text;
background-clip: text;
background-clip: text;
background-clip: text;
background-clip: text;
background-clip: text;
```

```
background-clip: text;
background-clip: text;
background-clip: text;
-webkit-text-fill-color: transparent;
}
.hero-subtitle {
font-size: 1.25rem;
color: var(--gray);
max-width: 700px;
margin-bottom: 2.5rem;
}
/* Detection Box */
.detection-box {
background-color: var(--white);
border-radius: 16px;
box-shadow: var(--shadow-lg);
padding: 2.5rem;
width: 100%;
max-width: 800px;
margin: 0 auto;
position: relative;
z-index: 1;
border: 1px solid var(--light-gray);
```

```
}
.detection-title {
font-size: 1.75rem;
font-weight: 600;
margin-bottom: 1.5rem;
text-align: center;
color: var(--dark);
}
.input-group {
margin-bottom: 1.5rem;
position: relative;
}
.input-label {
display: block;
margin-bottom: 0.75rem;
font-weight: 500;
color: var(--dark);
}
.url-input-container {
position: relative;
}
.url-input {
width: 100%;
```

```
padding: 1.25rem 1.5rem;
border: 2px solid var(--light-gray);
border-radius: 12px;
font-size: 1rem;
transition: var(--transition);
padding-left: 3rem;
background-color: var(--light);
}
.url-input:focus {
outline: none;
border-color: var(--primary);
box-shadow: 0 0 0 3px rgba(79, 70, 229, 0.2);
background-color: var(--white);
}
.url-icon {
position: absolute;
left: 1.25rem;
top: 50%;
transform: translateY(-50%);
color: var(--gray);
font-size: 1.25rem;
}
.check-btn {
```

```
width: 100%;
padding: 1.25rem;
background: linear-gradient(to right, var(--primary), var(--primary-light));
color: var(--white);
border: none;
border-radius: 12px;
font-size: 1rem;
font-weight: 600;
cursor: pointer;
transition: var(--transition);
display: flex;
align-items: center;
justify-content: center;
gap: 0.75rem;
box-shadow: var(--shadow);
}
.check-btn:hover {
background: linear-gradient(to right, var(--primary-dark), var(--primary));
transform: translateY(-2px);
box-shadow: var(--shadow-md);
}
.check-btn:active {
transform: translateY(0);
```

```
box-shadow: var(--shadow);
}
.loading {
display: none;
flex-direction: column;
align-items: center;
margin: 2rem 0;
animation: fadeIn 0.3s ease;
}
@keyframes fadeIn {
from { opacity: 0; }
to { opacity: 1; }
}
.spinner {
width: 3.5rem;
height: 3.5rem;
border: 4px solid rgba(79, 70, 229, 0.1);
border-radius: 50%;
border-top-color: var(--primary);
animation: spin 1s linear infinite;
margin-bottom: 1.5rem;
}
@keyframes spin {
```

```
to { transform: rotate(360deg); }
}
.loading-text {
font-size: 1.1rem;
color: var(--gray);
font-weight: 500;
}
.result {
display: none;
margin-top: 2rem;
padding: 2rem;
border-radius: 12px;
animation: fadeInUp 0.5s ease;
}
@keyframes fadeInUp {
from { opacity: 0; transform: translateY(20px); }
to { opacity: 1; transform: translateY(0); }
}
.result-header {
display: flex;
align-items: center;
margin-bottom: 1.5rem;
gap: 1rem;
```

```
}
.result-icon {
font-size: 2rem;
flex-shrink: 0;
}
.result-title {
font-size: 1.5rem;
font-weight: 600;
}
.result-text {
margin-bottom: 1.5rem;
color: var(--gray);
}
.result-details {
margin-top: 2rem;
}
.details-title {
font-weight: 600;
margin-bottom: 1rem;
font-size: 1.1rem;
color: var(--dark);
}
.details-list {
```

```
list-style: none;
background-color: var(--light);
border-radius: 8px;
overflow: hidden;
}
.detail-item {
display: flex;
justify-content: space-between;
padding: 1rem 1.5rem;
border-bottom: 1px solid var(--light-gray);
}
.detail-item:last-child {
border-bottom: none;
}
.detail-label {
font-weight: 500;
color: var(--gray);
}
.detail-value {
font-weight: 600;
text-align: right;
}
.safe {
```

```
background-color: rgba(16, 185, 129, 0.05);
border-left: 4px solid var(--success);
}
.phishing {
background-color: rgba(239, 68, 68, 0.05);
border-left: 4px solid var(--danger);
}
.suspicious {
background-color: rgba(245, 158, 11, 0.05);
border-left: 4px solid var(--warning);
}
/* Features Section */
.features {
padding: 6rem 0;
background-color: var(--white);
}
.section-title {
text-align: center;
font-size: 2.5rem;
font-weight: 700;
margin-bottom: 1rem;
color: var(--dark);
}
```

```
.section-subtitle {
text-align: center;
color: var(--gray);
max-width: 700px;
margin: 0 auto 4rem;
font-size: 1.1rem;
}
.features-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
gap: 2rem;
}
.feature-card {
background-color: var(--light);
border-radius: 16px;
padding: 2.5rem 2rem;
transition: var(--transition);
box-shadow: var(--shadow-sm);
border: 1px solid var(--light-gray);
text-align: center;
}
.feature-card:hover {
transform: translateY(-5px);
```

```
box-shadow: var(--shadow-lg);
border-color: var(--primary);
}
.feature-icon {
font-size: 2.5rem;
color: var(--primary);
margin-bottom: 1.5rem;
background: linear-gradient(to right, var(--primary), var(--primary-light));
-webkit-background-clip: text;
background-clip: text;
-webkit-text-fill-color: transparent;
}
.feature-title {
font-size: 1.5rem;
font-weight: 600;
margin-bottom: 1rem;
color: var(--dark);
}
.feature-text {
color: var(--gray);
line-height: 1.7;
}
/* Stats Section */
```

```
.stats {
padding: 5rem 0;
background: linear-gradient(135deg, var(--primary), var(--primary-dark));
color: var(--white);
position: relative;
overflow: hidden;
}
.stats::before {
content: '';
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
background-image: radial-gradient(circle at 20% 50%, rgba(255, 255, 255, 0.1) 0%, transparent 50%);
}
.stats-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
gap: 2rem;
text-align: center;
position: relative;
z-index: 1;
```

```
}
.stat-item {
padding: 2rem 1.5rem;
background-color: rgba(255, 255, 255, 0.1);
border-radius: 12px;
backdrop-filter: blur(5px);
transition: var(--transition);
}
.stat-item:hover {
transform: translateY(-5px);
background-color: rgba(255, 255, 255, 0.15);
}
.stat-number {
font-size: 3rem;
font-weight: 700;
margin-bottom: 0.5rem;
}
.stat-label {
font-size: 1.1rem;
opacity: 0.9;
}
/* Footer */
footer {
```

```
background-color: var(--dark);
color: var(--white);
padding: 5rem 0 2rem;
}
.footer-content {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
gap: 3rem;
margin-bottom: 3rem;
}
.footer-column h3 {
font-size: 1.5rem;
margin-bottom: 1.5rem;
position: relative;
padding-bottom: 1rem;
font-weight: 600;
}
.footer-column h3::after {
content: '';
position: absolute;
bottom: 0;
left: 0;
width: 50px;
```

```
height: 3px;
background-color: var(--primary);
}
.footer-links {
list-style: none;
}
.footer-link {
margin-bottom: 1rem;
}
.footer-link a {
color: rgba(255, 255, 255, 0.7);
text-decoration: none;
transition: var(--transition);
display: flex;
align-items: center;
gap: 0.5rem;
}
.footer-link a:hover {
color: var(--white);
padding-left: 5px;
}
.footer-link i {
font-size: 0.9rem;
```

```
}
.footer-text {
color: rgba(255, 255, 255, 0.7);
margin-bottom: 1.5rem;
line-height: 1.7;
}
.social-links {
display: flex;
gap: 1rem;
margin-top: 1.5rem;
}
.social-link {
display: flex;
align-items: center;
justify-content: center;
width: 40px;
height: 40px;
border-radius: 50%;
background-color: rgba(255, 255, 255, 0.1);
color: var(--white);
transition: var(--transition);
}
.social-link:hover {
```

```
background-color: var(--primary);
transform: translateY(-3px);
}
.footer-bottom {
text-align: center;
padding-top: 2rem;
border-top: 1px solid rgba(255, 255, 255, 0.1);
color: rgba(255, 255, 255, 0.5);
font-size: 0.9rem;
}
/* Responsive Design */
@media (max-width: 1024px) {
.hero-title {
font-size: 3rem;
}
}
@media (max-width: 768px) {
.nav-links {
display: none;
}
.mobile-menu-btn {
display: block;
}
```

```
.hero-title {
font-size: 2.5rem;
}
.hero-subtitle {
font-size: 1.1rem;
}
.detection-box {
padding: 2rem 1.5rem;
}
.detection-title {
font-size: 1.5rem;
}
.section-title {
font-size: 2rem;
}
}
@media (max-width: 480px) {
.hero {
padding: 4rem 0;
}
.hero-title {
font-size: 2rem;
}
```

```
.feature-card {
padding: 2rem 1.5rem;
}
.stat-number {
font-size: 2.5rem;
}
}
</style>
</head>
<body>
<!-- Header -->
<header>
<div class="container">
<nav class="navbar">
<a href="#" class="logo">
<i class="fas fa-shield-alt logo-icon"></i>
<span>PhishShield</span>
</a>
<div class="nav-links">
<a href="/" class="nav-link active">Home</a>
<a href="/how-it-works" class="nav-link">How It Works</a>
<a href="/about" class="nav-link">About</a>
<a href="/contact" class="nav-link">Contact</a>
```

```
</div>
<button class="mobile-menu-btn">
<i class="fas fa-bars"></i>
</button>
</nav>
</div>
</header>
<!-- Hero Section -->
<section class="hero">
<div class="container">
<div class="hero-content">
<h1 class="hero-title">Anonymity and Confidentiality in Website using Machine Learning</h1>
<p class="hero-subtitle">Protect yourself from malicious websites with our advanced phishing detection technology. Scan any URL instantly to check for potential threats.</p>
<div class="detection-box">
<h2 class="detection-title">Check a Website</h2>
<div class="input-group">
<label for="url-input" class="input-label">Enter URL to analyze</label>
<div class="url-input-container">
<i class="fas fa-link url-icon"></i>
<input type="url" id="url-input" class="url-input" placeholder="https://example.com" required>
</div>
</div>
<button id="check-btn" class="check-btn">
```

```
<i class="fas fa-search"></i>
Analyze URL
</button>
<div class="loading" id="loading">
<div class="spinner"></div>
<p class="loading-text">Scanning website for phishing indicators...</p>
</div>
<div class="result" id="result">
<div class="result-header">
<i class="fas fa-check-circle result-icon" id="result-icon"></i>
<h3 class="result-title" id="result-title">Result</h3>
</div>
<p class="result-text" id="result-text"></p>
<div class="result-details" id="result-details"></div>
</div>
</div>
</div>
</div>
</section>
<!-- Features Section -->
<section class="features">
<div class="container">
<h2 class="section-title">Advanced Protection Features</h2>
```

Our system uses multiple detection methods to identify phishing attempts with high accuracy

<h3 class="feature-title">URL Analysis</h3>

Examines URL structure for suspicious patterns, hidden characters, and known phishing techniques.

<h3 class="feature-title">Threat Intelligence</h3>

Checks against constantly updated databases of known phishing sites and malicious domains.

<h3 class="feature-title">Content Inspection</h3>

Analyzes page content for phishing indicators like fake login forms and suspicious scripts.

</section>

<!-- Stats Section -->

```
<section class="stats">
<div class="container">
<div class="stats-grid">
<div class="stat-item">
<div class="stat-number">93.5%</div>
<div class="stat-label">Detection Accuracy</div>
</div>
<div class="stat-item">
<div class="stat-number">10k</div>
<div class="stat-label">URLs Analyzed</div>
</div>
<div class="stat-item">
<div class="stat-number">24/7</div>
<div class="stat-label">Real-time Protection</div>
</div>
<div class="stat-item">
<div class="stat-number">100ms</div>
<div class="stat-label">Average Scan Time</div>
</div>
</div>
</div>
</section>
<!-- Footer -->
```

<footer>

<h3>PhishShield</h3>

Advanced phishing detection technology to protect users from malicious websites and online scams.

<h3>Quick Links</h3>

<ul class="footer-links">

<li class="footer-link"> Home

<li class="footer-link"> Data Set

<li class="footer-link"> How It Works <!-- Updated Link -->

<li class="footer-link"> Flask Documentation

<h3>Resources</h3>

<ul class="footer-links">

<li class="footer-link"> Blog

<li class="footer-link"> Documentation

<li class="footer-link"> Support

<li class="footer-link"> About <!-- Updated Link -->

<h3>Contact</h3>

<ul class="footer-links">

<li class="footer-link"> Email Us

<li class="footer-link"> Twitter

<li class="footer-link"> GitHub

<li class="footer-link"> Report Issue


```
</div>
</div>
<div class="footer-bottom">
<p>&copy; 2025 PhishShield. All rights reserved.</p>
</div>
</div>
</footer>
<script>
document.addEventListener('DOMContentLoaded', function() {
// Mobile menu toggle
const mobileMenuBtn = document.querySelector('.mobile-menu-btn');
const navLinks = document.querySelector('.nav-links');
mobileMenuBtn.addEventListener('click', function() {
navLinks.style.display = navLinks.style.display === 'flex' ? 'none' : 'flex';
});
// URL checking functionality
document.getElementById('check-btn').addEventListener('click', async function() {
const urlInput = document.getElementById('url-input').value.trim();
if (!urlInput) {
alert('Please enter a URL to check');
return;
}
// Validate URL format
```

```
try {
new URL(urlInput);
} catch (e) {
alert('Please enter a valid URL (e.g., https://example.com)');
return;
}
// Show loading indicator
document.getElementById('loading').style.display = 'flex';
document.getElementById('result').style.display = 'none';
try {
// Call your Flask backend API
const response = await fetch('http://localhost:5001/analyze', {
method: 'POST',
headers: {
'Content-Type': 'application/json',
},
body: JSON.stringify({ url: urlInput })
});
if (!response.ok) {
throw new Error(`HTTP error! status: ${response.status}`);
}
const analysisResult = await response.json();
// Display results
```

```
const resultDiv = document.getElementById('result');
const resultIcon = document.getElementById('result-icon');
const resultTitle = document.getElementById('result-title');
const resultText = document.getElementById('result-text');
const resultDetails = document.getElementById('result-details');
// Set appropriate styling and icons
if (analysisResult.isPhishing) {
resultDiv.className = 'result phishing';
resultIcon.className = 'fas fa-exclamation-triangle result-icon';
resultTitle.textContent = 'Phishing Detected!';
} else if (analysisResult.isSuspicious) {
resultDiv.className = 'result suspicious';
resultIcon.className = 'fas fa-question-circle result-icon';
resultTitle.textContent = 'Suspicious Website';
} else {
resultDiv.className = 'result safe';
resultIcon.className = 'fas fa-check-circle result-icon';
resultTitle.textContent = 'Safe Website';
}
resultText.textContent = analysisResult.message;
// Add detailed analysis if available
if (analysisResult.details) {
let detailsHtml = '<div class="details-title">Analysis Details</div><div class="details-list">';
```

```
for (const [feature, value] of Object.entries(analysisResult.details)) {
detailsHtml += `
<div class="detail-item">
<span class="detail-label">${feature}</span>
<span class="detail-value">${value}</span>
</div>
`;
}
detailsHtml += '</div>';
resultDetails.innerHTML = detailsHtml;
}
resultDiv.style.display = 'block';
} catch (error) {
alert('An error occurred while checking the URL. Please try again.');
console.error('Error:', error);
} finally {
document.getElementById('loading').style.display = 'none';
}
});
// Close mobile menu when clicking on a link
document.querySelectorAll('.nav-link').forEach(link => {
link.addEventListener('click', function() {
if (window.innerWidth <= 768) {
```

```
navLinks.style.display = 'none';
}
});
});
// Responsive adjustments
function handleResize() {
if (window.innerWidth > 768) {
navLinks.style.display = 'flex';
} else {
navLinks.style.display = 'none';
}
}
window.addEventListener('resize', handleResize);
handleResize();
});
</script>
</body>
</html>
```

- **about.html**

```
<!-- about.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>About Us | PhishShield</title>

<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap"
rel="stylesheet">

<style>

body { font-family: 'Poppins', sans-serif; background: #f9fafb; color: #1f2937; margin: 0; }

.container { max-width: 800px; margin: auto; padding: 2rem; }

h1, h2 { color: #4f46e5; margin-top: 2rem; }

p { margin: 1rem 0; line-height: 1.6; }

a { color: #4f46e5; text-decoration: none; }

a:hover { text-decoration: underline; }

</style>

</head>

<body>

<header style="background:#fff;box-shadow:0 1px 3px rgba(0,0,0,0.1);position:sticky;top:0;z-
index:100;">

<div class="container" style="display:flex;justify-content:space-between;align-
items:center;padding:1.5rem 0;">
```


 PhishShield

<nav style="display:flex;gap:2rem;">

Home

How It Works

About

Contact

</nav>

</header>

<main class="container">

<h1>About PhishShield</h1>

PhishShield is dedicated to protecting users from phishing threats using cutting-edge machine learning techniques. Our goal is to make the internet safer for everyone. We combine security research, data science, and practical experience to deliver a trustworthy solution for real-time phishing detection.

<h2>Abstract</h2>

A phishing attack is the simplest way to obtain sensitive information from innocent users. The aim of the phishers is to acquire critical information like username, password, and bank account details. Cybersecurity people are now looking for trustworthy and steady detection techniques for phishing website detection.

This project deals with machine learning technology for the detection of phishing URLs by extracting and analyzing various features of legitimate and phishing URLs.

Decision Tree, Random Forest, and Support Vector Machine algorithms are used to detect phishing websites. The aim of the paper is to detect phishing URLs as well as narrow down to the best machine learning algorithm by comparing accuracy rate, false positive rate, and false negative rate of each algorithm.

<h2>Phishing-Website-Detection</h2>

The financial industry involves an extremely high volume of real-time online transactions. This makes it vulnerable to fraud. Fraud detection uses security measures to prevent third parties from obtaining funds. This process involves a manual check and automated verification of transaction details to spot any unusual activity that may be a sign of attack and block it.

<h2>Email Phishing</h2>

Over the years, there have been many attacks of phishing and many people have lost huge sums of money by becoming victims. In a phishing attack, emails are sent to users claiming to be from legitimate organizations, asking for personal information like name, telephone, bank account number, or passwords. These emails direct the user to a fake website, where information is collected and used for illegal transactions.

<h2>Overview of the Project</h2>

Phishing is a luring technique used by attackers to exploit personal data. A phishing website mimics a legitimate one in appearance but is hosted at a different destination. Users unknowingly share data with these fake sites. Although several anti-phishing techniques have emerged, attackers continually develop ways to bypass them. Hence, there is a need for an efficient prediction mechanism.

This project is under construction and will be completed by May, 2025.

For more details, visit:

https://github.com/adityamishra-71/Anonymity-and-Confidentiality-in-Website-using-ML

</main>

<footer style="background:#1f2937;color:#fff;padding:2rem 0;text-align:center;margin-top:3rem;">

© 2025 PhishShield. All rights reserved.

</footer>

```
</body>

</html>
```

- **how-it-works.html**

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>How It Works | PhishShield</title>

<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap"
rel="stylesheet">

<style>

body { font-family: 'Poppins', sans-serif; background: #f9fafb; color: #1f2937; margin: 0; }

.container { max-width: 800px; margin: auto; padding: 2rem; }

h1 { color: #4f46e5; margin-bottom: 1rem; }

p { margin: 1rem 0; }

</style>

</head>

<body>

<header style="background:#fff;box-shadow:0 1px 3px rgba(0,0,0,0.1);position:sticky;top:0;z-
index:100;">

<div class="container" style="display:flex;justify-content:space-between;align-
items:center;padding:1.5rem 0;">
```


 PhishShield

<nav style="display:flex;gap:2rem;">

Home

How It Works

About

Contact

</nav>

</header>

<main class="container">

<h1>How It Works</h1>

How Does Phishing Work?

The easiest way to commit a robbery is probably to convince the victims they aren’t being robbed at all. That’s the phishing scammer’s basic model.

Phishing attacks begin with an email, phone call, SMS message, social media post, or the like that seems to be from a reputable source. From here, the attacker may have all sorts of end goals, such as tricking the victim into offering up account information, making a PayPal transfer, downloading disguised malware, and so on.

Let’s look at a common example. An attacker gets ahold of a victim’s email address or phone number. Then, the victim gets an email or text message from what appears to be their bank. The phishing message mentions an expiring special offer, suspected identity theft, or similar, and asks the victim to log in to their bank account. It links to a mock login webpage, and the victim unwittingly gives the attacker their login credentials.

The attack in this example, like most phishing attacks, carefully creates a sense of urgency that fools the victim into lowering their guard instead of taking time to consider whether the message is suspicious.

That may be more easily said than done, however, as there are quite a few tricks in the attacker's playbook.

Types of Phishing Attacks

Attackers have invented a wide variety of phishing techniques to exploit different technologies, trends, industries, and users. Here's a glance at some common types:

Email phishing: An email from a seemingly legitimate sender tries to trick the recipient into following a malicious link and/or downloading an infected file. The email address and any URL in a phishing email may use spoofing to appear legitimate.

Smishing/SMS phishing: Via text messages sent to mobile devices, attackers try to trick victims into giving out personal information, such as credit card numbers or other account numbers.

Vishing/Voice phishing: Essentially the same as smishing but carried out over a phone call, these attacks are after credit card information or other sensitive details.

Angler phishing: Posing as legitimate organizations on social media, attackers solicit personal information from victims, often by offering gift cards, discounts, etc.

Pop-up phishing: A common attack on Apple, Android, or other smartphones, an offer or warning message appears in a pop-up, generally containing a malicious link to trick victims into divulging personal data.

Spear phishing: While many phishing scams seek out victims at random, spear phishing attacks target specific individuals whose personal details the attacker already knows to some extent. This extra detail can greatly increase the odds of successful phishing.

Whaling attacks: Attackers phish executives or other important members of an organization in an attempt to obtain information that will give them privileged access to the target environment.

Clone phishing: Phishers send victims spoofed emails that seem to be from senders the victim trusts, such as financial institutions or accredited businesses like Amazon. This is closely related to spear phishing and a common tactic of business email compromise (BEC) attacks.

Evil twin phishing: Attackers lure victims with a trustworthy-looking Wi-Fi hotspot, and then carry out "man in the middle" attacks, intercepting data victims transfer over the connection.

Pharming: Attackers hijack the functionality of a Domain Name System (DNS) server so that it will redirect users to a malicious fake website even if they type a benign URL.

How Dangerous Are Phishing Attacks?

Phishing attacks can be extremely dangerous. Large phishing campaigns can affect millions of people, stealing sensitive data, planting ransomware and other malware as well as gaining access to the most sensitive areas of a company’s systems.

Loss of sensitive data such as financial information, reputational damage, and regulatory issues are among the many possible consequences of a successful phishing attack at the organizational level. Risks for any phishing victim can include loss or compromise of sensitive data, and organizations also face possible reputational damage and regulatory issues.

How Does Phishing Affect Businesses?

At the organizational level, the consequences of a successful phishing attack can be far-reaching and serious. Financial losses can stem from a compromised corporate bank account. Data loss can stem from phishing that leads to a ransomware attack. An organization can sustain major reputational damage from any breach of sensitive data that necessitates public disclosure.

Furthermore, any of these can have even more serious consequences in turn. Cybercriminals may sell stolen data on the dark web, including to unscrupulous competitors. On top of that, many breaches will need to be disclosed to industry or government regulatory bodies that may levy fines or other sanctions. It may even involve the organization in cybercrime investigations, which can be time-consuming and attract negative attention.

How Do I Protect My Organization Against Phishing Attacks?

Fortunately, most types of phishing can be stopped if you take the right precautions. That means:

Use effective cybersecurity countermeasures. Modern antivirus and anti-phishing solutions, alongside effective spam filters, will screen out many phishing attempts.

Keep operating systems and browsers up to date. Software providers regularly address newfound vulnerabilities in their products, without which your system will be left exposed.

Protect data with automatic backups. Implement a regular process of system data backup so that you can recover in the event of a breach.

Use advanced multifactor authentication (MFA). Zero trust strategies such as MFA create additional layers of defense between attackers and your internal systems.

Ensure your users are educated. Cybercriminals constantly invent new strategies, and email security won’t catch everything. Your users and your organization at large will be safer if all users understand how to identify suspicious email messages and report phishing.

What Are the Signs of Phishing?

When it comes to phishing, the safest users are the ones who know how to avoid getting hooked. While a short summary no substitute for focused security awareness training, here are a few key warning signs of attempted phishing:

Discrepancies in domain names: Email addresses and web domains may have inconsistencies. For example, if you receive an email claiming to be from a well-known brand, the email address may not match it.

Spelling errors: Although phishing attacks have become much more effective, the messages still often contain spelling or grammatical mistakes.

Unfamiliar greetings: Sometimes, the style of a greeting or signoff can be a clue that something isn’t right. Take note of someone who always opens messages with “Hi!” suddenly says “Dear friend” instead.

Short and sweet: Phishing emails often keep information sparse, relying on ambiguity to throw off victims’ judgment. If too many important details are missing, it may be a sign of a phishing attempt.

Unusual requests: An email asking you to do something unusual, especially without explanation, is a big red flag. For example, a phishing attempt could claim to be from your IT team, asking you to download a file without specifying a reason.

Phishing and AI

AI-driven phishing attacks leverage AI tools to enhance the sophistication and effectiveness of phishing campaigns. AI automates and personalizes various aspects of the attack process, making phishing even more challenging to detect. For example, chatbots are commonly used to craft highly convincing, error-free phishing emails. What’s more, attackers are increasingly harnessing advanced AI services such as deepfake technology and voice cloning to impersonate reputable organizations or people and deceive victims. They exploit various communication channels, including emails, phone and video calls, SMS, and encrypted messaging applications.

Generative AI is rapidly driving the phishing threat landscape forward, enabling automation and efficiency across numerous stages of the attack chain. By rapidly analyzing publicly available data, such as

details about organizations or executives, GenAI saves threat actors time in targeting, so they can facilitate more precise, targeted attacks. By eliminating spelling errors and grammatical mistakes, GenAI tools enhance the credibility of phishing communications. What's more, GenAI can quickly create sophisticated phishing pages or extend its capabilities to generate malware and ransomware for secondary attacks. As GenAI tools and tactics rapidly evolve, phishing attacks will become more dynamic (and challenging to detect) by the day.

The growing popularity and use of GenAI tools like ChatGPT and Drift is already beginning to impact phishing activity and the rise of AI-driven attacks. Countries like the US and India, where these tools are highly utilized according to ThreatLabz research in the 2024 AI Security Report, are top targets for phishing scams and face the highest number of encrypted attacks in the past year, a subset of which are phishing attacks.

Phishing Protection with Zscaler

Because it relies on exploiting human nature to succeed, user compromise is one of the most difficult security challenges to overcome. To detect active breaches and minimize the damage successful breaches can cause, you need to implement effective phishing prevention controls as part of a broader zero trust strategy.

The Zscaler Zero Trust Exchange™ platform, built on a holistic zero trust architecture to minimize the attack surface, prevent compromise, eliminate lateral movement, and stop data loss, helps stop phishing by:

Preventing attacks: Features like full TLS/SSL inspection, browser isolation, and policy-driven access control prevent access from malicious websites.

Preventing lateral movement: Once in your system, malware can spread, causing even more damage. With the Zero Trust Exchange, users connect directly to apps, not your network, so malware can't spread from them.

Stopping insider threats: Our cloud proxy architecture stops private app exploit attempts and detects even the most sophisticated attack techniques with full inline inspection.

Stopping data loss: The Zero Trust Exchange inspects data in motion and at rest to prevent potential data theft from an active attacker.

</main>

<footer style="background:#1f2937;color:#fff;padding:2rem 0;text-align:center;margin-top:3rem;">

© 2025 PhishShield. All rights reserved.

</footer>

</body>

</html>

- **contact.html**

<!-- contact.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Contact Us | PhishShield</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap"
rel="stylesheet">

<style>

body { font-family: 'Poppins', sans-serif; background: #f9fafb; color: #1f2937; margin: 0; }

.container { max-width: 1200px; margin: auto; padding: 2rem; }

.team-grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); gap: 1.5rem; }

.member { padding: 1rem; border: 1px solid #e5e7eb; border-radius: 8px; background: #f3f4f6; text-align: center; }

.member h3 { margin: 0.5rem 0; }

```
.socials a { margin: 0 0.5rem; color: #4f46e5; font-size: 1.2rem; text-decoration: none; }

.institution { margin-top: 3rem; text-align: center; font-size: 0.95rem; color: #4b5563; line-height: 1.6; }

</style>

</head>

<body>

<header style="background:#fff;box-shadow:0 1px 3px rgba(0,0,0,0.1);position:sticky;top:0;z-index:100;">

<div class="container" style="display:flex;justify-content:space-between;align-items:center;padding:1.5rem 0;">

<a href="/" style="font-size:1.5rem;font-weight:700;text-decoration:none;color:#4338CA;display:flex;align-items:center;gap:0.75rem;">

<i class="fas fa-shield-alt" style="font-size:1.75rem;color:#4F46E5;"></i> PhishShield

</a>

<nav style="display:flex;gap:2rem;">

<a href="/" class="nav-link">Home</a>

<a href="/how-it-works" class="nav-link">How It Works</a>

<a href="/about" class="nav-link active">About</a>

<a href="/contact" class="nav-link">Contact</a>

</nav>

</div>

</header>

<main class="container">

<h1>Contact Our Team</h1>

<p>You can reach out to any of our team members through the links below:</p>

<div class="team-grid">
```

```
<div class="member">

<h3>Aditya Mishra</h3>

<div class="socials">

<a href="https://github.com/adityamishra-71"><i class="fab fa-github"></i></a>

<a href="mailto:adityamishra.am71@gmail.com"><i class="fas fa-envelope"></i></a>

<a href="https://www.linkedin.com/in/aditya-mishra-a771201b6/"><i class="fab fa-linkedin"></i></a>

</div>

</div>

<div class="member">

<h3>Anshuman Soni</h3>

<div class="socials">

<a href="https://github.com/anshumanji"><i class="fab fa-github"></i></a>

<a href="mailto:anshumansoni4546@gmail.com"><i class="fas fa-envelope"></i></a>

<a href="https://www.linkedin.com/in/anshuman-soni-
587001224?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_a
pp"><i class="fab fa-linkedin"></i></a>

</div>

</div>

<div class="member">

<h3>Anuj Mishra</h3>

<div class="socials">

<a href="https://github.com/Anujmishra7069"><i class="fab fa-github"></i></a>

<a href="mailto:anujmishra7069@gmail.com"><i class="fas fa-envelope"></i></a>

<a href="https://www.linkedin.com/in/anuj-mishra-816003213/"><i class="fab fa-linkedin"></i></a>
```

```
</div>

</div>

<div class="member">

<h3>Shivam Singh</h3>

<div class="socials">

<a href="https://github.com/shivamsinghAIMLops32"><i class="fab fa-github"></i></a>

<a href="mailto:shivamsinghcse19@gmail.com"><i class="fas fa-envelope"></i></a>

</div>

</div>

<div class="member">

<h3>Mr. UmaShanker (Guide)</h3>

<div class="socials">

<a href="mailto:umashanker.aiml@gniot.net.in"><i class="fas fa-envelope"></i></a>

</div>

</div>

</div>

<div class="institution">

<p>Members (1-4): GNIOT (Engineering Institute), Department of Computer Science (AI&ML)</p>

<p>Guide: Mr. UmaShankar</p>

<p>Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Uttar Pradesh, India</p>

<p>Emails: anshumansoni4546@gmail.com, adityamishra.am71@gmail.com,
anujmishra7069@gmail.com, shivamsinghcse19@gmail.com, umashanker.aiml@gniot.net.in</p>

</div>

</main>
```

```
<footer style="background:#1f2937;color:#fff;padding:2rem 0;text-align:center;margin-top:3rem;">
<div class="container">
<p style="margin:0;color:rgba(255,255,255,0.6);">&copy; 2025 PhishShield. All rights reserved.</p>
</div>
</footer>
</body>
</html>
```

- **layout.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>{% block title %}{% endblock title %}</title>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.css" />
</head>
<body>
<div class="container">
{% include 'includes/_navbar.html'%}
{% block body %}{% endblock body%}
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/js/bootstrap.min.js"></script>
```

```
</div>

</body>

</html>
```

6.2. BACK END PROGRAM:

- **RandomForestModel.py**

```
import pandas as pd

import numpy as np

import pickle

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

# ☑ Load Dataset

legitimate_urls = pd.read_csv(r"C:\Aditya\Study\Phishing-Website-Detection-
master\extracted_csv_files\legitimate-urls.csv")

phishing_urls = pd.read_csv(r"C:\Aditya\Study\Phishing-Website-Detection-
master\extracted_csv_files\phishing-urls.csv")

# ☑ Merge Data

urls = pd.concat([legitimate_urls, phishing_urls], ignore_index=True)

# ☑ Drop Unnecessary Columns (Modify If Needed)

urls = urls.drop(columns=["Unnamed: 0"], errors="ignore")

# ☑ Separate Features & Labels

X = urls.drop(columns=["label"]) # Features

y = urls["label"] # Target (0 = Legitimate, 1 = Phishing)
```

☑ Handle Categorical Columns

encoders = {}

for col in X.select_dtypes(include=['object']).columns:

encoders[col] = LabelEncoder()

X[col] = encoders[col].fit_transform(X[col])

☑ Split Data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

☑ Train Model

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)

☑ Save Model & Encoders

pickle.dump(rf_model, open(r"C:\Aditya\Study\Phishing-Website-Detection-master\Phishing website detection using UI\RandomForestModel.sav", "wb"))

pickle.dump(encoders, open(r"C:\Aditya\Study\Phishing-Website-Detection-master\Phishing website detection using UI\encoders.sav", "wb"))

print("☑ Model & Encoders saved successfully!")

- **Classifier.py**

coding: utf-8

In[1]:

import pandas as pd

Collection of Data

In[2]:

legitimate_urls = pd.read_csv("legitimate-urls.csv")

```
phishing_urls = pd.read_csv("phishing-urls.csv")

# In[3]:

print(len(legitimate_urls))

print(len(phishing_urls))

# ## Data PreProcessing

# #### Data is in two data frames so we merge them to make one dataframe

# Note: two dataframes has same column names

# In[4]:

urls = legitimate_urls.append(phishing_urls)

# In[5]:

urls.head(5)

# In[6]:

print(len(urls))

print(urls.columns)

# #### Removing Unnecessary columns

# In[7]:

urls = urls.drop(urls.columns[[0,3,5]],axis=1)

print(urls.columns)

# #### Since we merged two dataframes top 1000 rows will have legitimate urls and bottom 1000 rows will have phishing urls. So if we split the data now and create a model for it will overfit or underfit so we need to shuffle the rows before splitting the data into training set and test set

# In[8]:

# shuffling the rows in the dataset so that when splitting the train and test set are equally distributed

urls = urls.sample(frac=1).reset_index(drop=True)
```

```
# #### Removing class variable from the dataset

urls_without_labels = urls.drop('label',axis=1)

urls_without_labels.columns

labels = urls['label']

#labels

# #### splitting the data into train data and test data

import random

random.seed(100)

from sklearn.model_selection import train_test_split

data_train, data_test, labels_train, labels_test = train_test_split(urls_without_labels, labels, test_size=0.20,
random_state=100)

print(len(data_train),len(data_test),len(labels_train),len(labels_test))

print(labels_train.value_counts())

print(labels_test.value_counts())

# #### Checking the data is split in equal distribution or not

"""

train_0_dist = 711/1410

print(train_0_dist)

train_1_dist = 699/1410

print(train_1_dist)

test_0_dist = 306/605

print(test_0_dist)

test_1_dist = 299/605

print(test_1_dist)
```

```
"""
# ## Random Forest
from sklearn.ensemble import RandomForestClassifier
RFmodel = RandomForestClassifier()
RFmodel.fit(data_train,labels_train)
rf_pred_label = RFmodel.predict(data_test)
#print(list(labels_test)),print(list(rf_pred_label))
from sklearn.metrics import confusion_matrix,accuracy_score
cm2 = confusion_matrix(labels_test,rf_pred_label)
print(cm2)
print(accuracy_score(labels_test,rf_pred_label))
"""
# Saving the model to a file
import pickle
file_name = "RandomForestModel.sav"
pickle.dump(RFmodel,open(file_name,'wb'))
"""
```

- **Classifier2.py**

```
# coding: utf-8
# In[1]:
import pandas as pd
# ## Collection of Data
# In[2]:
```

```
legitimate_urls = pd.read_csv("legitimate-urls.csv")

phishing_urls = pd.read_csv("phishing-urls.csv")

# In[3]:

legitimate_urls.head(10)

phishing_urls.head(10)

# ## Data PreProcessing

# #### Data is in two data frames so we merge them to make one dataframe

# Note: two dataframes has same column names

# In[4]:

urls = legitimate_urls.append(phishing_urls)

# In[5]:

urls.head(5)

# In[6]:

urls.columns

# #### Removing Unnecessary columns

# In[7]:

urls = urls.drop(urls.columns[[0,3,5]],axis=1)

# #### Since we merged two dataframes top 1000 rows will have legitimate urls and bottom 1000 rows will have phishing urls. So if we split the data now and create a model for it will overfit or underfit so we need to shuffle the rows before splitting the data into training set and test set

# In[8]:

# shuffling the rows in the dataset so that when splitting the train and test set are equally distributed

urls = urls.sample(frac=1).reset_index(drop=True)

# #### Removing class variable from the dataset
```

```
# In[9]:

urls_without_labels = urls.drop('label',axis=1)

urls_without_labels.columns

labels = urls['label']

#labels

# #### splitting the data into train data and test data

# In[49]:

from sklearn.model_selection import train_test_split

data_train, data_test, labels_train, labels_test = train_test_split(urls_without_labels, labels, test_size=0.30,
random_state=100)

# In[37]:

print(len(data_train),len(data_test),len(labels_train),len(labels_test))

# In[151]:

print(labels_train.value_counts())

print(labels_test.value_counts())

# #### Checking the data is split in equal distribution or not

# In[158]:

train_0_dist = 711/1410

print(train_0_dist)

train_1_dist = 699/1410

print(train_1_dist)

test_0_dist = 306/605

print(test_0_dist)

test_1_dist = 299/605
```

```
print(test_1_dist)

# #### creating the model and fitting the data into the model

# In[50]:

from sklearn.tree import DecisionTreeClassifier

DTmodel = DecisionTreeClassifier(random_state=0)

DTmodel.fit(data_train,labels_train)

# #### predicting the result for test data

# In[51]:

pred_label = model.predict(data_test)

# In[52]:

print(pred_label),print(list(labels_test))

# #### creating confusion matrix and checking the accuracy

# In[54]:

from sklearn.metrics import confusion_matrix,accuracy_score

cm = confusion_matrix(labels_test,pred_label)

print(cm)

accuracy_score(labels_test,pred_label)

# ## Random Forest

# In[55]:

from sklearn.ensemble import RandomForestClassifier

RFmodel = RandomForestClassifier()

RFmodel.fit(data_train,labels_train)

# In[56]:
```

```
rf_pred_label = rfModel.predict(data_test)

# In[57]:

print(list(labels_test)),print(list(rf_pred_label))

# In[58]:

cm2 = confusion_matrix(labels_test,rf_pred_label)

cm2

# In[60]:

accuracy_score(labels_test,rf_pred_label)

# ### Improving the efficiency

# In[138]:

imp_rf_model = RandomForestClassifier(n_estimators=100,max_depth=30,max_leaf_nodes=10000)

# In[140]:

imp_rf_model.fit(data_train,labels_train)

# In[142]:

imp_pred_label = imp_rf_model.predict(data_test)

# In[144]:

cm3 = confusion_matrix(labels_test,imp_pred_label)

cm3

# In[146]:

accuracy_score(labels_test,imp_pred_label)
```

- **FeatureExtraction.py**

```
# coding: utf-8
import pandas as pd
from urllib.parse import urlparse
import re
from bs4 import BeautifulSoup
import whois
import urllib.request
import time
import socket
from urllib.error import HTTPError
from datetime import datetime
class FeatureExtraction:
def __init__(self):
pass
def getProtocol(self, url):
return urlparse(url).scheme
def getDomain(self, url):
return urlparse(url).netloc
def getPath(self, url):
return urlparse(url).path
def havingIP(self, url):
"""If the domain part has IP then it is phishing, otherwise legitimate."""
```

```
match = re.search(
'(([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\.'
'([01]?\d\d?|2[0-4]\d|25[0-5])\.([01]?\d\d?|2[0-4]\d|25[0-5])\/)|' # IPv4
'((0x[0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})\.(0x[0-9a-fA-F]{1,2})\/)' # IPv4 in hex
'(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url) # IPv6
return 1 if match else 0 # 1 = phishing, 0 = legitimate
def long_url(self, url):
"""Classifies URLs based on length."""
if len(url) < 54:
return 0 # Legitimate
elif 54 <= len(url) <= 75:
return 2 # Suspicious
else:
return 1 # Phishing
def have_at_symbol(self, url):
"""Checks if '@' symbol is present in URL."""
return 1 if "@" in url else 0 # 1 = phishing, 0 = legitimate
def redirection(self, url):
"""Checks for '//' in the URL path (indicates possible redirection)."""
return 1 if "//" in urlparse(url).path else 0 # 1 = phishing, 0 = legitimate
def prefix_suffix_separation(self, url):
"""Checks if domain contains '-' (common in phishing URLs)."""
return 1 if "-" in urlparse(url).netloc else 0 # 1 = phishing, 0 = legitimate
```

```
def sub_domains(self, url):
"""Classifies URLs based on the number of subdomains."""
dot_count = url.count(".")
if dot_count < 3:
return 0 # Legitimate
elif dot_count == 3:
return 2 # Suspicious
else:
return 1 # Phishing
def shortening_service(self, url):
"""Detects URL shorteners (bit.ly, tinyurl, etc.), commonly used in phishing."""
match = re.search(
'bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
'db\.tt|qr\.ae|adf\.ly|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|q\.gs|is\.gd|po\.st|'
'bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|x\.co|'
'prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|'
'v\.gd|tr\.im|link\.zip\.net', url)
return 1 if match else 0 # 1 = phishing, 0 = legitimate
def web_traffic(self, url):
"""This function is disabled because Alexa API no longer works."""
```

```
return 2 # Default "suspicious" value
def domain_registration_length(self, url):
"""Checks domain registration length."""
try:
domain_info = whois.whois(urlparse(url).netloc)
expiration_date = domain_info.expiration_date
if expiration_date is None:
return 1 # Phishing
today = datetime.today()
if isinstance(expiration_date, list):
expiration_date = expiration_date[0]
registration_length = (expiration_date - today).days
return 1 if registration_length / 365 <= 1 else 0 # 1 = phishing, 0 = legitimate
except:
return 1 # Assume phishing if WHOIS lookup fails
def age_domain(self, url):
"""Checks the age of the domain."""
try:
domain_info = whois.whois(urlparse(url).netloc)
creation_date = domain_info.creation_date
if creation_date is None:
return 1 # Phishing
today = datetime.today()
```

```
if isinstance(creation_date, list):
creation_date = creation_date[0]
age = (today - creation_date).days
return 1 if (age / 30) < 6 else 0 # 1 = phishing, 0 = legitimate
except:
return 1 # Assume phishing if WHOIS lookup fails
def dns_record(self, url):
"""Checks if a DNS record exists."""
try:
domain_info = whois.whois(urlparse(url).netloc)
return 0 if domain_info else 1 # 0 = legitimate, 1 = phishing
except:
return 1 # Phishing (if lookup fails)
def https_token(self, url):
"""Detects misuse of HTTPS in the URL."""
match = re.search('https://|http://', url)
try:
if match.start(0) == 0:
url = url[match.end(0):]
return 1 if re.search('http|https', url) else 0
except:
return 1
return 0
```

```
def getAttributess(url):
"""Extracts all features from a given URL."""
fe = FeatureExtraction()
features = {
'Protocol': fe.getProtocol(url),
'Domain': fe.getDomain(url),
'Path': fe.getPath(url),
'Having_IP': fe.havingIP(url),
'URL_Length': fe.long_url(url),
'Having_@_symbol': fe.have_at_symbol(url),
'Redirection_//_symbol': fe.redirection(url),
'Prefix_suffix_separation': fe.prefix_suffix_separation(url),
'Sub_domains': fe.sub_domains(url),
'Tiny_URL': fe.shortening_service(url),
'Web_Traffic': fe.web_traffic(url),
'Domain_Registration_Length': fe.domain_registration_length(url),
'DNS_Record': fe.dns_record(url),
'Age_Domain': fe.age_domain(url),
'HTTPS_Token': fe.https_token(url)
}
return pd.DataFrame([features])
```

- **app.py**

```
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
import FeatureExtraction
import pickle
import numpy as np
import pandas as pd
# ☑ Load Model & Encoders
with open("RandomForestModel.sav", "rb") as model_file:
RFmodel = pickle.load(model_file)
with open("encoders.sav", "rb") as encoder_file:
encoders = pickle.load(encoder_file)
# ☑ Extract Feature Names from Model Training
feature_names = RFmodel.feature_names_in_
app = Flask(__name__)
CORS(app)
# ☑ Routes
@app.route('/')
def index():
return render_template("home.html")
@app.route('/about')
def about():
return render_template("about.html")
```

```
@app.route('/how-it-works')
def how_it_works():
return render_template("how-it-works.html")
@app.route('/contact')
def contact():
return render_template("contact.html")
@app.route('/getURL', methods=['POST'])
def getURL():
if request.method == 'POST':
url = request.form['url']
print(f"Checking URL: {url}")
# ☑ Extract Features
data = FeatureExtraction.getAttributess(url)
# ☑ Convert DataFrame to List (Fixes Shape Issues)
if isinstance(data, pd.DataFrame):
data = data.iloc[0].tolist()
elif isinstance(data, np.ndarray):
data = data.flatten().tolist()
elif not isinstance(data, list):
print("❌ ERROR: Feature extraction failed! Unexpected data type.")
return render_template("home.html", error="Feature extraction failed!")
# ☑ Validate Feature Count
if len(data) != len(feature_names):
```

print(f"❌ ERROR: Feature count mismatch! Expected {len(feature_names)}, got {len(data)}")

return render_template("home.html", error="Feature extraction failed!")

☑ Convert Data to DataFrame

df = pd.DataFrame([data], columns=feature_names)

☑ Encode Categoricals

for col in df.columns:

if df[col].dtype == "object":

if col in encoders:

df[col] = encoders[col].transform(df[col].astype(str))

else:

df[col] = df[col].astype("category").cat.codes

☑ Clean Data

df.fillna(0, inplace=True)

df = df.astype(np.float64)

☑ Predict

predicted_value = RFmodel.predict(df)

result = "Legitimate" if predicted_value == 0 else "Phishing"

return render_template("home.html", error=result)

@app.route('/analyze', methods=['POST'])

def analyze():

data = request.get_json()

url = data.get('url')

if not url:

```
return jsonify({'message': 'No URL provided', 'isPhishing': False, 'isSuspicious': False}), 400

# ☑ Extract Features

features = FeatureExtraction.getAttributess(url)

if isinstance(features, pd.DataFrame):

features = features.iloc[0].tolist()

elif isinstance(features, np.ndarray):

features = features.flatten().tolist()

elif not isinstance(features, list):

return jsonify({'message': 'Feature extraction failed!', 'isPhishing': False, 'isSuspicious': True}), 500

if len(features) != len(feature_names):

return jsonify({'message': 'Feature count mismatch!', 'isPhishing': False, 'isSuspicious': True}), 500

features = [x.item() if hasattr(x, "item") else x for x in features]

df = pd.DataFrame([features], columns=feature_names)

for col in df.columns:

if df[col].dtype == "object":

if col in encoders:

df[col] = encoders[col].transform(df[col].astype(str))

else:

df[col] = df[col].astype("category").cat.codes

df.fillna(0, inplace=True)

df = df.astype(np.float64)

predicted_value = int(RFmodel.predict(df)[0])

details = {k: (v.item() if hasattr(v, "item") else v) for k, v in zip(feature_names, features)}
```

```
if predicted_value == 1:
return jsonify({
'message': 'Phishing website detected!',
'isPhishing': True,
'isSuspicious': False,
'details': details
})
else:
return jsonify({
'message': 'This website appears safe.',
'isPhishing': False,
'isSuspicious': False,
'details': details
})
if __name__ == "__main__":
app.run(debug=True, port=5001)
```

Chapter 7

Results and Discussion

7.1. Results:

The project focuses on detecting phishing websites using machine learning techniques, specifically evaluating the effectiveness of feature-based classification.

1. Feature Extraction
    a. A total of 14 features were extracted from URLs, including length, domain age, web traffic rank, and subdomain analysis, ensuring a comprehensive analysis of phishing behavior.
    b. Datasets of legitimate and phishing URLs were combined and preprocessed, removing inconsistencies to ensure reliable model training.
2. Model Training and Testing
    a. The dataset was split into 70% training and 30% testing subsets to maintain model integrity.
    b. The Random Forest Classifier outperformed Decision Tree and SVM algorithms in identifying patterns within the feature set.
3. Feature Importance Analysis
    a. Key features such as URL length and web traffic rank emerged as the most impactful for classification.
    b. A bar plot visually represented feature rankings, providing insights into their relative contributions to prediction accuracy.
4. Performance Metrics
    a. Comparative analysis of algorithms highlighted significant differences in false positive and false negative rates, emphasizing the need for balanced evaluation.
    b. The Random Forest Classifier demonstrated robust results, reflecting its capacity for handling feature-rich datasets.

7.1.1. Comparative Analysis based on Accuracy

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Remarks
Decision Tree	85.3	83.4	84.2	83.7	Easy to interpret but prone to overfitting
Support Vector Machine (SVM)	89.8	88.3	87.6	87.7	High accuracy but computationally expensive
Random Forest	**93.5**	**92.0**	**91.8**	**91.9**	Best overall performance with strong generalization

Table. 7.1.1.i:- Classifier's Performance

7.2. DISCUSSION:

1. Model Accuracy and Relevance
    a. The implemented models effectively distinguished phishing from legitimate URLs, leveraging well-defined feature extraction techniques.
    b. The study underscored the importance of specific features, like domain age and HTTPS presence, in enhancing detection reliability.
2. Scalability and Robustness
    a. The system demonstrated scalability, processing large datasets without compromising efficiency.
    b. The Random Forest model's adaptability and ability to manage overfitting were pivotal in ensuring consistent performance.
3. Challenges and Limitations
    a. The reliance on static URL features posed challenges in detecting sophisticated, dynamically generated phishing attacks.

b. Real-world applications may require periodic updates to feature sets as phishing tactics evolve.

4. Future Enhancements

    a. Integration of dynamic behavioral analysis, such as page content scrutiny, to complement URL-based detection.

    b. Development of real-time monitoring systems to deploy the detection mechanism in active cybersecurity frameworks.

This comprehensive exploration validates the potential of machine learning in phishing detection, setting a foundation for future advancements in cybersecurity.

Chapter 8

SCREEN SHOTS

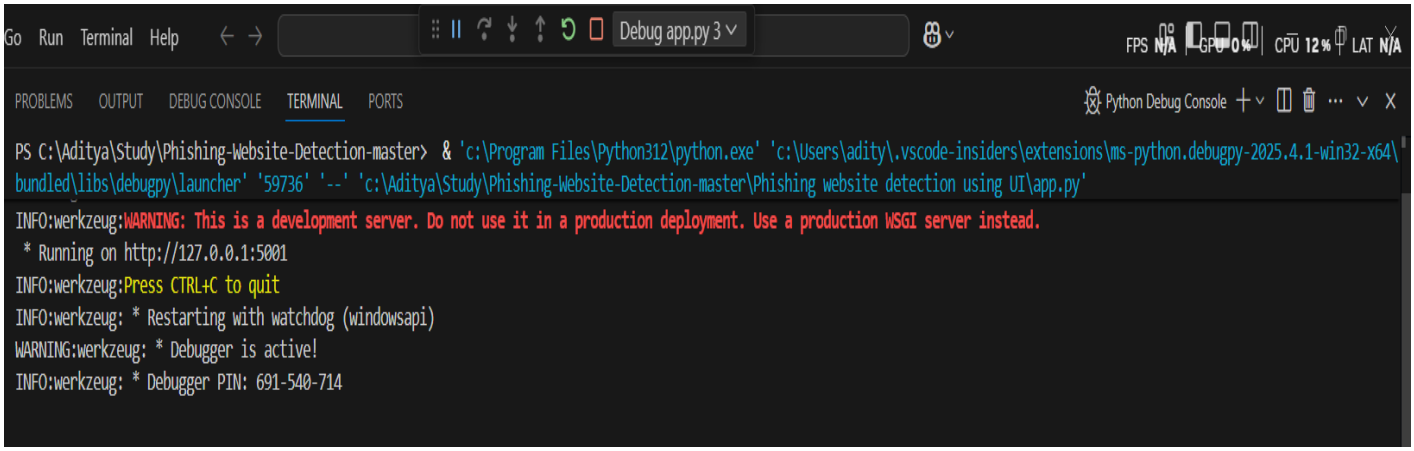


Fig.8.1. Output and link to open local webapp using Flask

FRONT END:

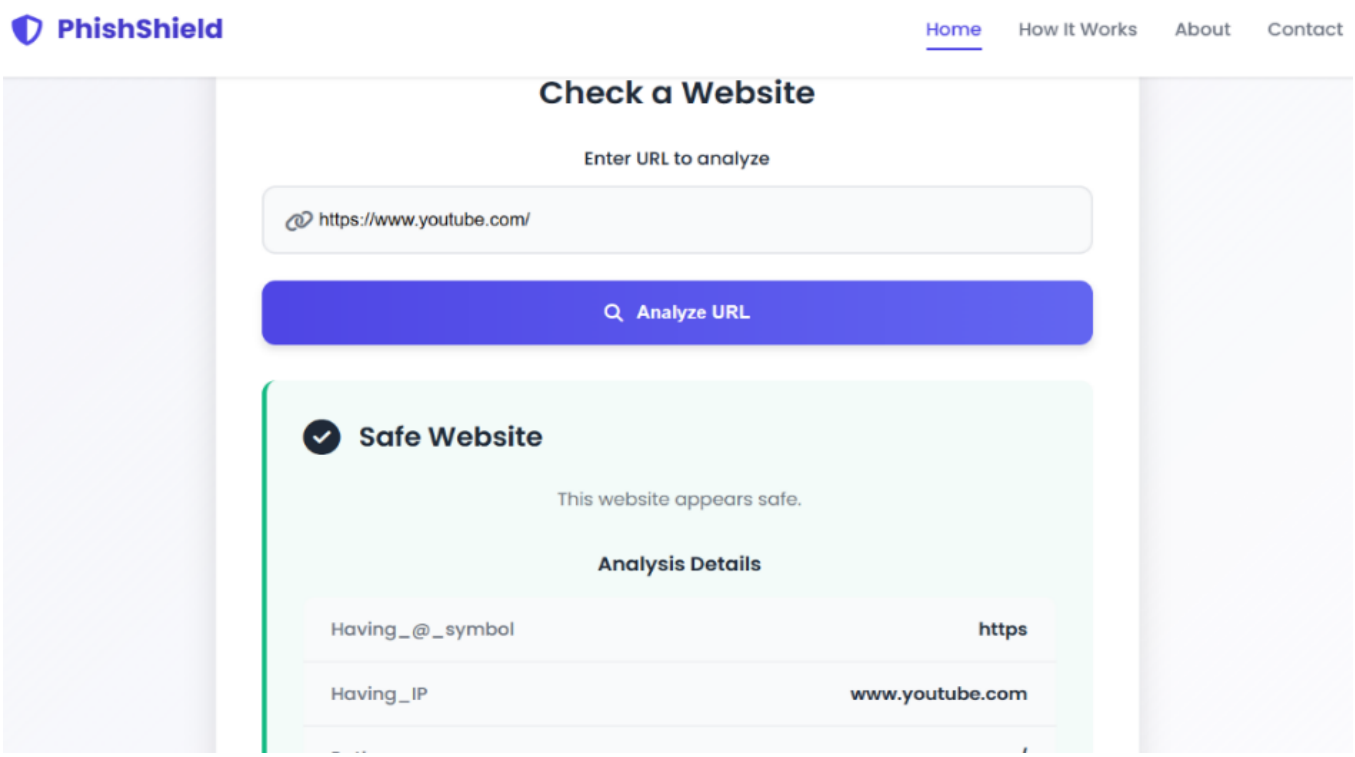


Fig.8.2. Youtube URL as legitimate website

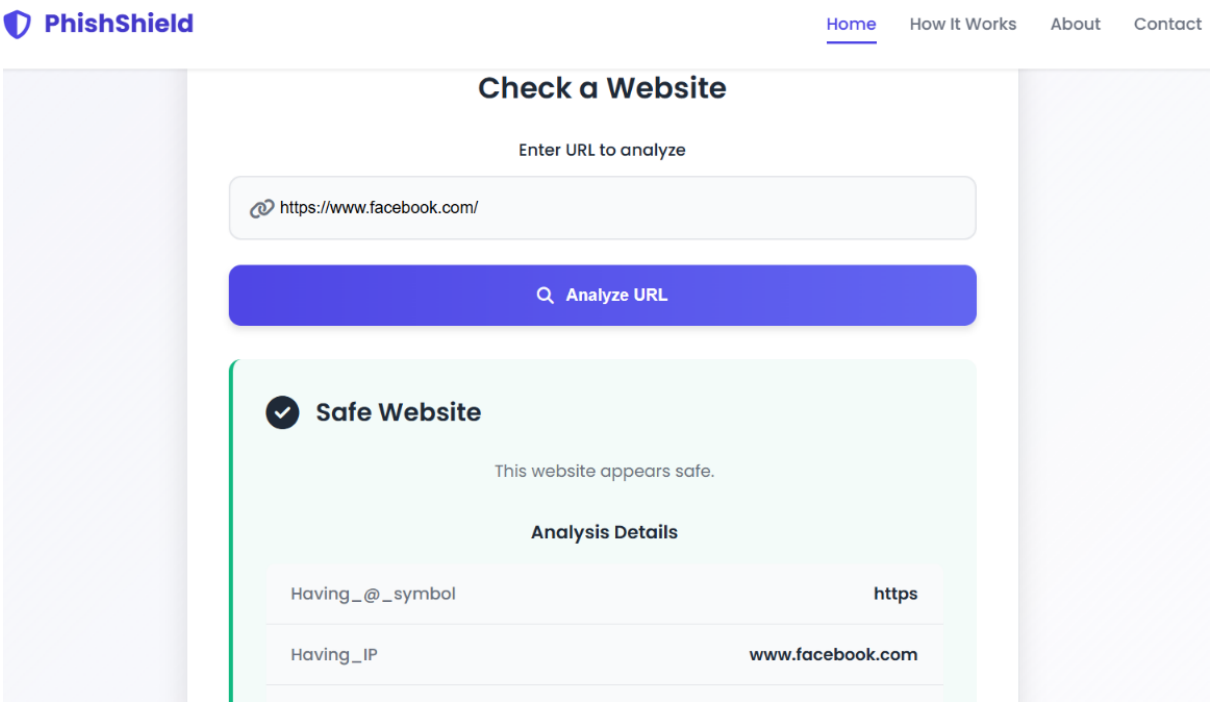


Fig.8.3. Facebook URL as legitimate website

Phishing Websites Examples :-

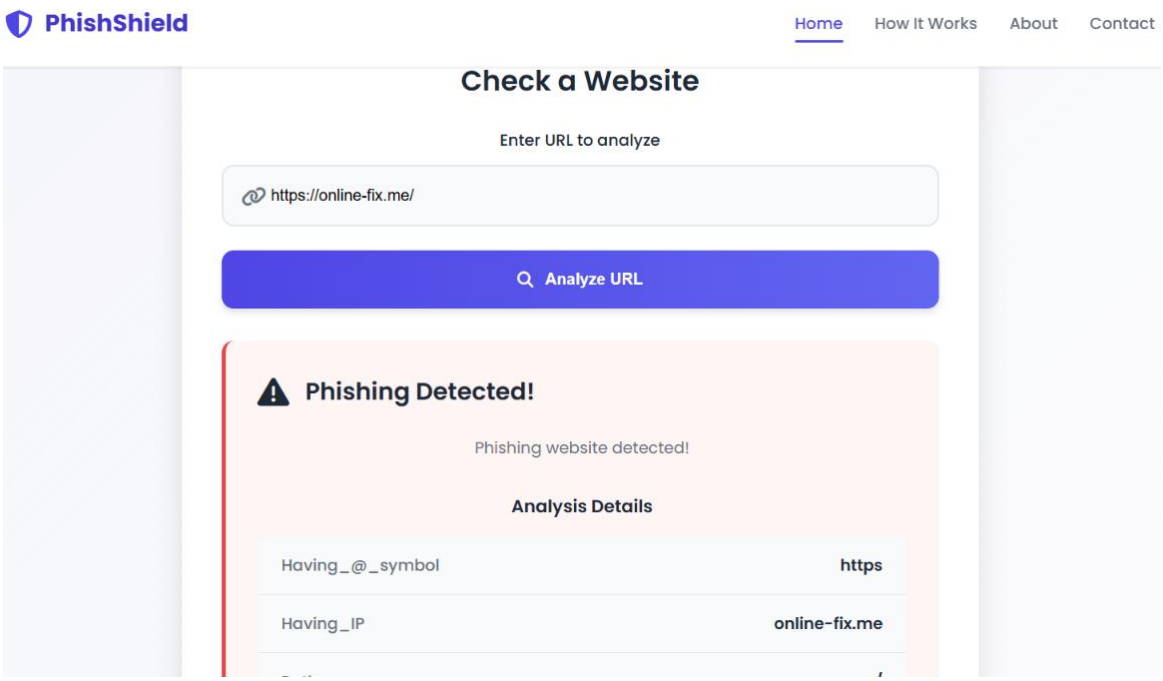


Fig.8.4. OnlineFix(3rd party gaming website) URL as Phishing website

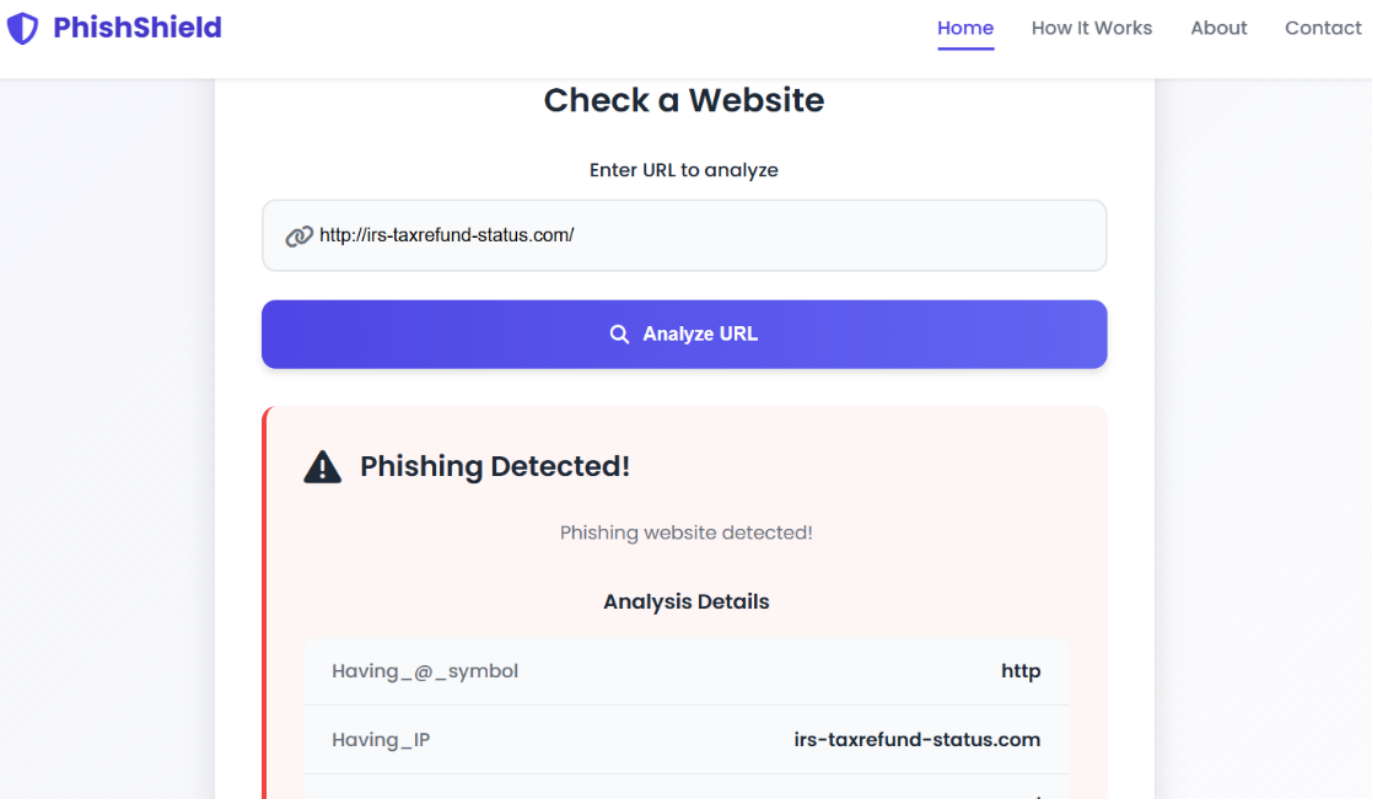


Fig.8.5. Irs taxrefund fake website URL as Phishing website

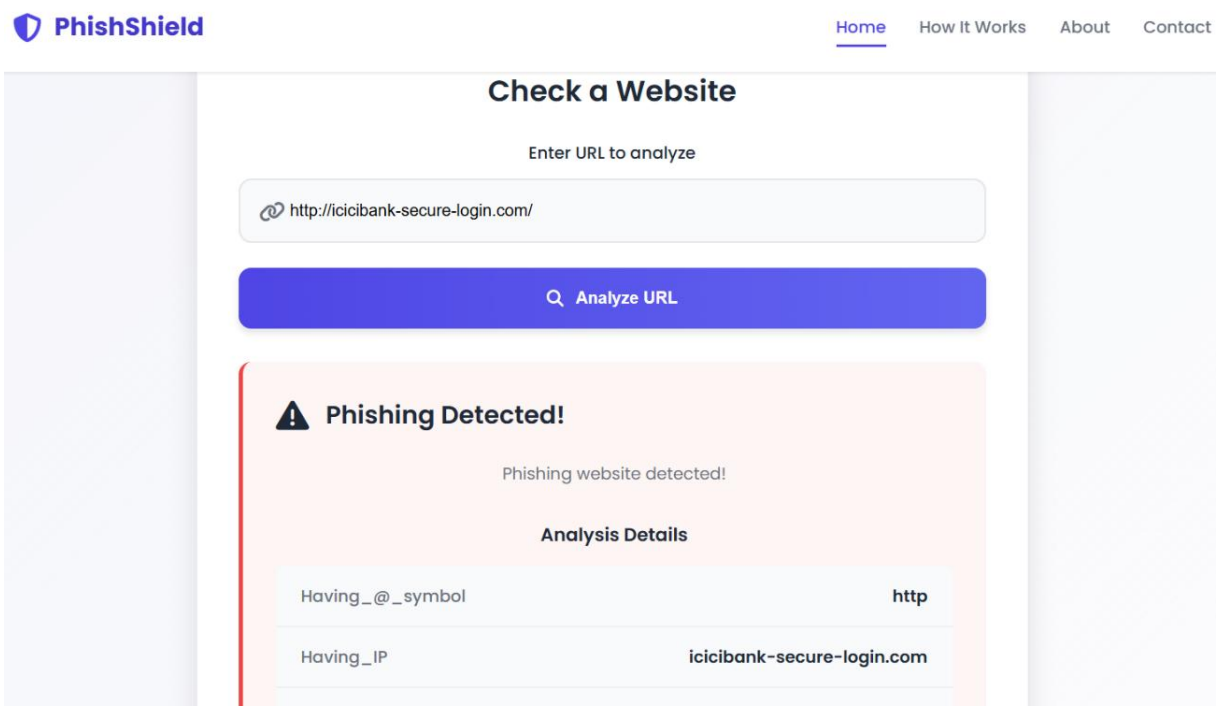


Fig.8.6. Icici bank fake website URL as Phishing website

Comparison with McAfee Web Advisor:

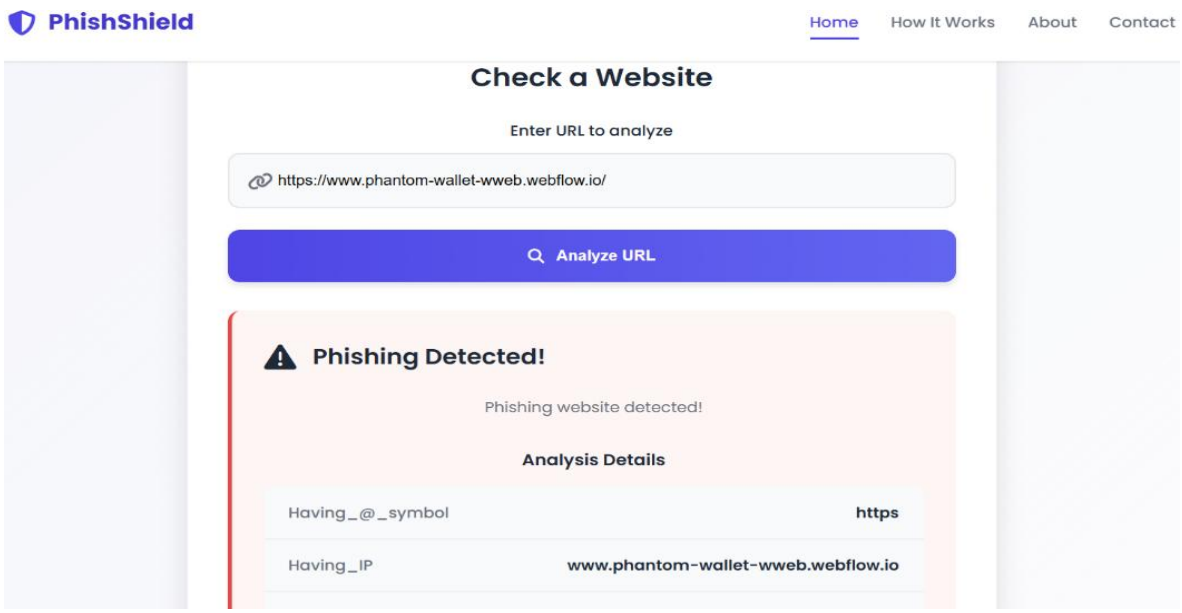


Fig.8.7. Icici bank fake website URL as Phishing website

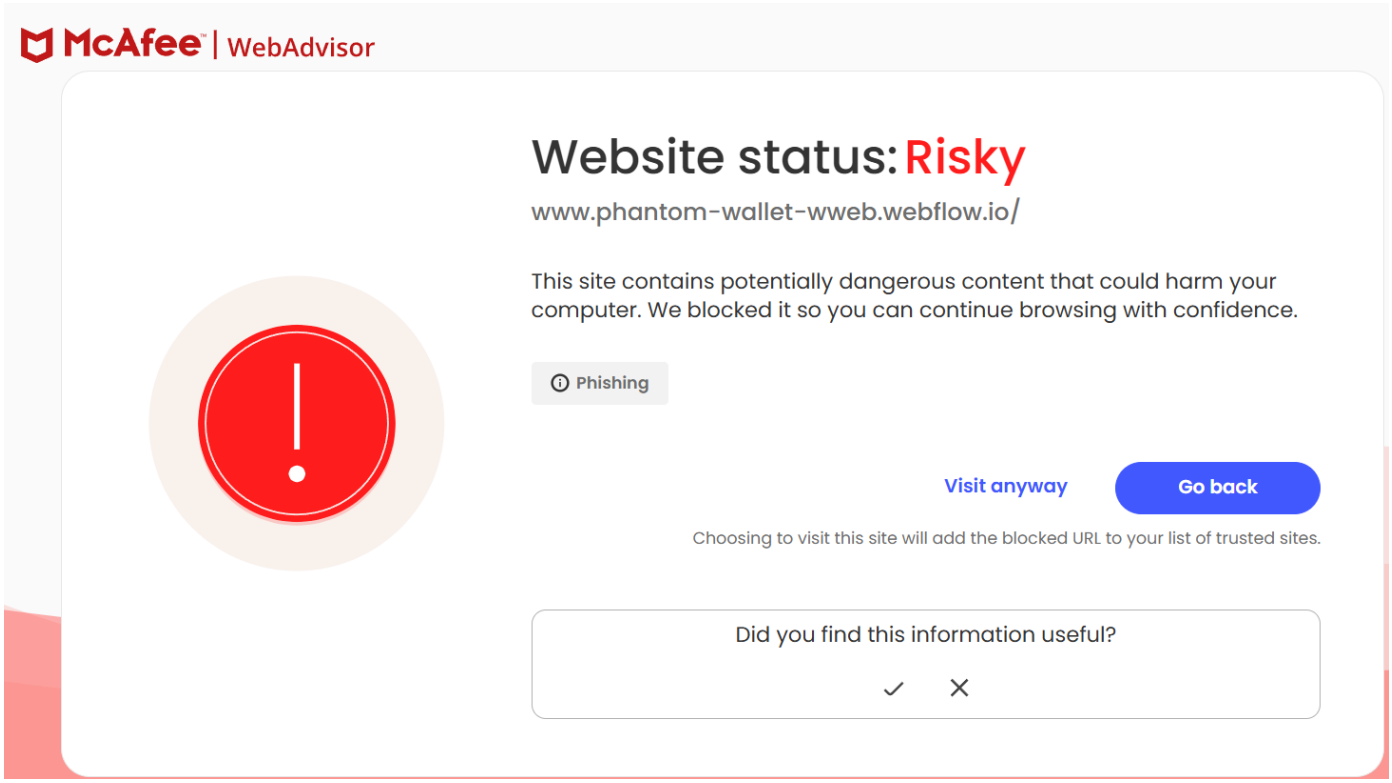


Fig.8.8. McAfee web advisor's warning

Chapter 9

Future Scope and Conclusion

9.1. Conclusion :

Random Forest demonstrates the best performance in phishing website detection because it reaches the highest accuracy rate among machine learning algorithms. This model achieves 93.5% accuracy while maintaining effective generalization ability which makes it an appropriate tool for real-world deployment. The high computational requirements as well as complex system implementation challenges make Support Vector Machine (SVM) an excellent choice for a 89.8% accurate system. Decision Tree provides straightforward implementation together with easy interpretation but its accuracy rate at 85.3% stands below the other models while it contains overfitting vulnerabilities. Random Forest stands out as the top choice for phishing website identification because it maintains appropriate precision while reaching high accuracy alongside excellent recall results. The future development potential of feature selection techniques should be investigated to achieve better detection accuracy by using deep learning methods.

9.2. Future Scope :

The phishing detection system developed in this project has shown promising results in identifying malicious websites using machine learning techniques. However, there is still significant scope for improvement and expansion. Future work will focus on the following areas:

1. **Real-time Detection and Automation**
    a. Enhancing the system to perform real-time phishing detection with minimal latency.
    b. Automating dataset updates to continuously train the model on the latest phishing techniques and emerging threats.
2. **Browser Extension and API Development**
    a. Developing a browser extension that can provide instant alerts to users when visiting a suspicious website.
    b. Creating an API that allows third-party applications and security platforms to integrate phishing detection capabilities.
3. **Hybrid Approach for Better Accuracy**
    a. Combining machine learning with rule-based techniques (such as blacklists and heuristics) to reduce false positives and improve reliability.

b. Implementing ensemble learning methods that combine multiple models for higher detection accuracy.

4. **Cross-Platform Compatibility**
    a. Expanding the system to work across different devices and platforms, including mobile applications.
    b. Optimizing resource usage for better performance on low-power devices.
5. **Adversarial Attack Defense Mechanisms**
    a. Strengthening the model against adversarial attacks where attackers modify URLs or page content to bypass detection.
    b. Implementing robust feature selection methods to counter phishing techniques that evolve over time.

By implementing these future improvements, the phishing detection system can become more effective, adaptive, and user-friendly, ultimately enhancing cybersecurity for individuals and organizations.

REFERENCES

[1] Marwa Abd Al Hussein Qasim, Dr. Nahla Abbas Flayh, “Phishing Website Detection Using Machine Learning: A Review” June 2023Wasit Journal of Pure sciences 2(2):270-2812(2):270-281

[2] Rishikesh Mahajan, Irfan Siddavatam ,”Phishing Website Detection using Machine Learning Algorithms” , International Journal of Computer Applications, Volume 181 - Number 23 Year of Publication: 2018

[3] S. A. Anwekar and V. Agrawal, "PHISHING WEBSITE DETECTION USING MACHINE LEARNING ALGORITHMS."

[4] S. Jain, "Phishing Websites Detection Using Machine Learning," Available at SSRN 4121102.

[5] A. Lakshmanarao, P. S. P. Rao, and M. B. Krishna, "Phishing website detection using novel machine learning fusion approach," in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021: IEEE, pp. 1164-1169.

[6] L. Tang and Q. H. Mahmoud, "A Deep Learning-Based Framework for Phishing Web-site Detection," IEEE Access, vol. 10, pp. 1509-1521, 2021.

[7] A. D. Kulkarni and L. L. Brown III, "Phishing websites detection using machine learn-ing," 2019.

[8] I. Tyagi, J. Shad, S. Sharma, S. Gaur, and G. Kaur, "A novel machine learning ap-proach to detect phishing websites," in 2018 5th International conference on signal pro-cessing and integrated networks (SPIN), 2018: IEEE, pp. 425-430.

[9] M. Karabatak and T. Mustafa, "Performance comparison of classifiers on reduced phishing website dataset," in 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018: IEEE, pp. 1-5.

[10] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G.-G. Geng, "Boosting the phishing detection performance by semantic analysis," in 2017 IEEE international conference on big data (big data), 2017: IEEE, pp. 1063-1070.

[11] W. Fadheel, M. Abusharkh, and I. Abdel-Qader, "On Feature selection for the prediction of phishing websites," in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2017: IEEE, pp. 871-876.

[12] A. A. Ahmed and N. A. Abdullah, "Real time detection of phishing websites," in 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Con-ference (IEMCON), 2016: IEEE, pp. 1-6.

[13] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," EURASIP Journal on Information Security, vol. 2016, no. 1, pp. 1-11, 2016.

[14] M. Aydin and N. Baykal, "Feature extraction and classification phishing websites based on URL," in 2015 IEEE Conference on Communications and Network Security (CNS), 2015: IEEE, pp. 769-770.