ADAPTIVE PHISHING DEFENSE: A CHROME EXTENSION POWERED BY MLSELM FOR REAL-TIME URL MONITORING

A PROJECT REPORT

Submitted by

FATHIMA JEMINA M (210181601014)

LAKSHMI PRIYA S (210181601022)

Under the guidance of

Dr. Sharmila Sankar

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

(CYBER SECURITY)

APRIL 2025

B.S. Abdur Rahman ™
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

BONAFIDE CERTIFICATE

Certified that this project report "**ADAPTIVE PHISHING DEFENSE: A CHROME EXTENSION POWERED BY MLSELM FOR REAL-TIME URL MONITORING**" is the bonafide work of **FATHIMA JEMINA M (21801601014) and LAKSHMI PRIYA S(210181601022)** who carried out the project work under my supervision. Certified further, that to the best of our knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. SHARMILA SANKAR
SUPERVISOR
Dean
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science & Technology
Vandalur, Chennai – 600 048

SIGNATURE
Dr. W. AISHA BANU
HEAD OF THE DEPARTMENT
Professor
Department of CSE
B. S. Abdur Rahman Crescent
Institute of Science & Technology
Vandalur, Chennai – 600 048

B.S. Abdur Rahman ™
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

VIVA VOCE EXAMINATION

The viva voce examination of the **CSD 4201,** Project work titled **"ADAPTIVE PHISHING DEFENSE: A CHROME EXTENSION POWERED BY MLSELM FOR REAL-TIME URL MONITORING",** submitted by **FATHIMA JEMINA M (21801601014)** and **LAKSHMI PRIYA S(210181601022)** is held on_____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely express our heartfelt gratitude to **Prof. Dr. T. MURUGESAN**, Vice chancellor and **Dr. N. THAJUDDIN**, Pro-Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology, for providing us an environment to carry out our project successfully.

We sincerely thank **Dr. N. RAJA HUSSAIN**, Registrar, for furnishing every essential facility for doing our project.

We thank **Dr. SHARMILA SANKAR**, Dean, School of Computer, Information and Mathematical Sciences for her motivation and support.

We thank **Dr. W. AISHA BANU**, Professor and Head, Department of Computer Science and Engineering, for providing strong oversight of vision, strategic direction and valuable suggestions.

We express our sincere thanks to the Project Review Committee members, from the Department of Computer Science and Engineering, **Dr. N. SABIYATH FATIMA**, Professor and **DR. R. AKILA**, Associate Professor for their valuable suggestions and support.

We obliged our project supervisor **Dr. SHARMILA SANKAR**, **Dean**, School of Computer, Information and Mathematical Sciences for her professional guidance and continued assistance during our project.

We thank class advisor and project coordinator **Dr. C. HEMA, Associate Professor**, Department of Computer Science and Engineering for her professional guidance and continued assistance throughout the project period.

We thank all **the Faculty members** and the **System Staff** of the Department of Computer Science and Engineering for their valuable support and assistance at various stages of project development.

FATHIMA JEMINA M

LAKSHMI PRIYA S

ABSTRACT

Phishing continues to be a very serious security risk with which the end users are manipulated for revealing sensitive information. Phishing attacks have become highly dynamic, and traditional anti-phishing measures have failed to counter these threats; thus, intelligent defense system is required. Strong implementation is needed in reaching or in real-time detection of phishing attack using machine learning (ML) techniques over analysing Uniform Resource Locators (URLs). The main objective thus is to develop a Multilayer Stacked Ensemble Learning Model (MLSELM) which would come for real-time URL monitoring so that it can easily tell if it is a legitimate or malicious URL.

The enhanced Phishing detection system uses Random Forest and Xtreme gradient Boosting (XGBoost) classifiers along with MLSELM to achieve better and better results in phishing detection. Multiple models within allows for the learning process to become more accurate in its classification results using different data sets. The project methodology consists of three major stages: dataset preparation, feature selection and model training, which are combined with a Chrome extension for real-time detection.

During implementation, phishing URLs and legitimate URLs are all collected for logging purposes, followed by a process of pre-processing and feature extraction to aid detection. The MLSELM model is then trained to best identify the phishing URLs. A Chrome extension is provided to alert users in real time and secure from phishing attacks.

Therefore, it can be said that with an accuracy rate of 95% to as high as 97%, the phishing detection system can easily detect phishing URLs. It offers opportunities for MLSELM in real-time adaptive defence against phishing threats. Securing cyberspace is safeguarding web applications from the cyber-attacks that threaten to grow for individuals and organizations. It aims to continuously evolve to cope with the new tricks introduced by changing forms of phishing.

TABLE OF CONTENTS

CHAPTER NO		TITLE	PAGE NO.
		ABSTRACT	**iv**
		TABLE OF CONTENTS	**v**
		LIST OF ABBREVIATIONS	**viii**
		LIST OF FIGURES	**ix**
1		**INTRODUCTION**	**1**
1.1		OVERVIEW	1
	1.1.1	MULTI-LAYER STACKED ENSEMBLE LEARNING MODEL	2
1.2		OBJECTIVE	3
1.3		DESCRIPTION	3
1.4		ORGANIZATION OF THE REPORT	3
2		**LITERATURE SURVEY**	**5**
3		**PROBLEM DEFINITION AND METHODOLOGIES**	**10**
3.1		PROBLEM DEFINITION	10
3.2		EXISTING SYSTEM	10
3.3		PROPOSED SYSTEM	11
3.4		METHODOLOGIES	12
	3.4.1	DATA COLLECTION	13
	3.4.2	FEATURE EXTRACTION	14
	3.4.3	MACHINE LEARNING MODEL DEVELOPMENT	15
	3.4.4	MODEL TRAINING AND EVALUATION	16
	3.4.5	CHROME EXTENSION DEVELOPMENT	17
	3.4.6	BACKEND SERVER INTEGRATION	18

		3.4.7	PHISHING DETECTION DEPLOYMENT	19
4			**DESIGN PROCESS**	**21**
	4.1		DESIGN OVERVIEW	21
	4.2		DATA FLOW DIAGRAM	24
	4.3		ARCHITECTURE DIAGRAM	25
		4.3.1	USER INTERFACE	26
		4.3.2	FRONT-END	26
		4.3.3	BACK-END	26
	4.4		PROJECT REQUIREMENTS	27
5			**IMPLEMENTATION**	**30**
	5.1		PHISHING DETECTION	30
		5.1.1	DATA SOURCING	31
		5.1.2	FEATURE VECTORIZATION	32
	5.2		TRAINING THE PHISHING MODEL	33
		5.2.1	RANDOM FOREST	34
		5.2.2	XGBOOST	35
		5.2.3	ENSEMBLE MODEL	36
	5.3		CHROME EXTENSION DEVELOPMENT	38
	5.4		INTEGRATION OF PHISHING MODEL WITH CHROME EXTENSION	39
		5.4.1	FLASK SERVER	40
6			**RESULTS AND ANALYSIS**	**42**
	6.1		EVALUATION METRICS	42
	6.2		PHISHING MODEL	43
		6.2.1	MODEL OBSERVATIONS	44
	6.3		CHROME EXTENSION	50
	6.4		INTEGRATION OF PHISHING MODEL	50
	6.5		RESULTS AND DISCUSSION	51

7	CONCLUSION AND FUTURE WORK	54
	REFERENCES	55
	APPENDIX	57
	A1 – SOURCE CODE	57
	A2 – SCREENSHOTS	71
	TECHNICAL BIOGRAPHY	75

LIST OF ABBREVIATIONS

MLSELM	MULTI-LAYER STACKED ENSEMBLE LEARNING MODEL
ML	MACHINE LEARNING
URL	UNIFORM RESOURCE LOCATOR
HTTPS	HYPER TEXT TRANSFER PROTOCOL SECURITY
CSS	CASCADING STYLE SHEETS
HTML	HYPER TEXT MARKUP LANGUAGE
TF-IDF	TERM FREQUENCY – INVERSE DOCUMENT FREQUENCY
XGBOOST	EXTREME GRADIENT BOOSTING
CNN	CONVOLUTIONAL NEURAL NETWORK
RNN	RECURRENT NEURAL NETWORK
BRNN	BI-DIRECTIONAL RECURRENT NEURAL NETWORK
PCA	PRINCIPLE COMPONENT ANALYSIS
ANN	ARTIFICAL NEURAL NETWORK
RESMLP	RESIDUAL MULTI LAYER PERSEPTRON

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE.NO
4.1	Design Process for Phishing Detection System	22
4.2	Data Flow for Phishing Detection System	24
4.3	Architecture Diagram for Phishing Detection System	25
6.1	Feature Extraction Model Comparison	45
6.2	ML Model Comparison	46
6.3	Confusion Matrix	47
6.4	Precision	48
6.5	Recall	49
6.6	F1-Score	49
6.7	Chrome Extension detecting safe website	52
6.8	Chrome Extension detecting phishing website	52
6.9	Chrome Extension detecting safe website	53
6.10	Chrome Extension detecting phishing website	53
A2. 1	PhiUSIIL_Phishing_URL_Dataset	71
A2. 2	Phishing_Dataset	71
A2. 3	Phishing Detection Prediction 1	72
A2. 4	Phishing Detection Prediction 2	72
A2. 5	Phishing Detection Prediction 3	72

A2. 6	Phishing Detection Prediction 4	72
A2. 7	Chrome Extension Prediction 1	73
A2. 8	Chrome Extension Prediction 2	73
A2. 9	Chrome Extension Prediction 3	74
A2.10	Chrome Extension Prediction 4	74

CHAPTER 1

INTRODUCTION

The phishing detection system monitors all URLs continuously in order to detect malicious sites immediately while users are browsing. It will immediately alert the users on the harmful links, preventing them from clicking to avoid giving away sensitive information. It provides a secure defense to both private and corporate users, offering extra security to protect sensitive information. Thus, it guarantees safe and effective browsing protection for all people.

1.1 OVERVIEW

Cyber security entails the essential safeguarding of critical information as well as online assets as an integral part of security systems. Advances in web services alongside changing technology have introduced more sinister and dishonest methods of phishing. One of the most common cyber threats includes phishing that results in perpetrators accessing users' confidential information such as passwords and credit card details. Phishing attacks inflict massive financial loss and allow identity forgery in addition to inflicting serious data breaches on all types of users and institutions.

Data protection by digital means through Information Security practices secures data against unauthorized abuse along with cutting off and securing it from theft. The biggest battle of information security is to detect phishing sites using fraudulent means to replicate legitimate sites. The rule-based systems and blacklists adhere to conventional approaches that are ineffective in identifying new or zero-day phishing attacks since cybercriminals continually adjust their attack techniques. Global reports show that phishing operations attack every country globally with over 72,000 organizations in 145 business sectors and more. EasyDMARC's anti-phishing system blocked over 190 million attacks in 2023 which shows the extent of the problem.

The existence of machine learning (ML) and deep learning (DL) models was to bypass these system limitations. The DEPHIDES phishing tools and PhishHaven utilize DL methods for classifying URLs and Multi-Layer Stacked Ensemble Learning (MLSELM) combines multiple classifiers in an attempt to

improve solution quality. Utilizing these models involves large datasets with costly computing resources.

The proposal essentially relies upon machine learning classifiers to automatically identify phishing with respect to URL legitimacy. Text features are extracted using Term Frequency-Inverse Document Frequency (TF-IDF), and the ensemble framework follows upon application of Random Forest and XGBoost. The methodology is perfect in sniffing phishing URLs with minimal chances of falsely ascribing a known legitimate site to be phish. It further builds up new signatures for phishing without much reliance on traditional blacklisting methods.

The adaptive system launches proactive real-time phishing detection features that improve cybersecurity with protection of sensitive information and build scalable countermeasures.

1.1.1 MULTI-LAYER STACKED ENSEMBLE LEARNING MODEL

The MLSELM system employs different models to identify phishing attempts through URL assessments. By adapting its learning feature the system provides real-time protection which enables users to detect complex phishing threats effectively.

Benefits of MLSELM:

- MLSELM enhances phishing detection precision through its multiple machine learning model approach which achieves better results during practical applications.
- The system adjusts to new phishing approaches through its ability to learn continuously from updated information for maintaining current threat identification abilities.
- The users obtain instant alerts about possible phishing URLS through the integrated Chrome extension system of the model.

Drawbacks of MLSELM:

- The effective utilization of MLSELM needs continual maintenance of extensive phishing and legitimate URL datasets although this presents challenges to the system administrator.

- Large quantities of data and advanced processing methods resulting from the ensemble approach demand higher system resources during both training sessions and real-time operations.

1.2 OBJECTIVE

Deployment of an advanced phishing attack detection system that integrates MLSELM with providing for real-time monitoring of phishing activities with a Chrome extension for user notification. It is very important for the system to accurately detect phishing URLs. The system will have to always learn new approaches to phishing. The users should be able to access their browser easily and securely. Enhanced online security reduced the threats one is exposed to through phishing.

1.3 DESCRIPTION

It aims to develop an adaptive phishing detection system through MLSELM model which builds a Chrome extension which will be integrated to monitor URLs. Accurate URL analysis, domain reputation checks, and assessment of the characteristics of the webpage, identifies attempts of phishing within the system. Notification is done once the user enters the URL. With these ensemble learning methods on board, the system as well will be on the same accuracy levels as it can be while detecting fresh phishing URLs. Through continued training and hyper-tuning of the MLSELM model, the system has become more learned in dealing with present-day phishing attacks. Various cybersecurity solutions like online banking, corporate security functions, and seamless browsing for an individual are offered by the system. On the efficiency scale along with the proper measures of accuracy, it comes under false detections minimized.

1.4 ORGANIZATION OF THE REPORT

Chapter 1 discusses the project overview. The main function of Adaptive Phishing Defense uses MLSELM within a Chrome extension to keep track of URLs in real-time for phishing detection. The chapter describes the project while aligning it to the system requirements and proposed design.

Chapter 2 the review section evaluates previously published research papers about phishing detection whereas it illustrates both their implementation strategies and their encountered issues.

Chapter 3 introduces a problem statement along with user demands and practical phishing situations. The section presents a summary of active phishing detection approaches alongside their weaknesses along with a description of the proposed solution and its proposed workflow which offsets previous technique limitations.

Chapter 4 focuses on the complete design process of the project. The design process gets explained through sequential instructions which utilize Data Flow Diagrams and Architecture Diagrams. The section includes specifications about necessary hardware infrastructure as well as software requirements.

Chapter 5 covers the implementation steps of the project. The document illustrates which methods and tools functioned to create the phishing detector extension for Chrome. The system components receive detailed explanations together with clear visual examples for better understanding.

Chapter 6 demonstrates the essential project outcomes from the research. This section addresses the questions collected during requirements gathering and shows results from analyzing the phishing detection system performance. The research delivers recommendations alongside detection accuracy improvement strategies in addition to guidelines for dealing with upcoming phishing attacks.

Chapter 7 delivers a project summary and explains potential areas of improvement for the future. The system proves its capability to detect phishing URLs in real time but requires further work to develop better feature selection along with model optimization and resistance against evolving cyber threats.

CHAPTER 2

LITERATURE SURVEY

Kalabarige lakshmana rao et al.[1] developed by to detect phishing websites, a multilayer stacked ensemble learning model whose accuracy percentages 97.32%. The model designs architecturally Multi-Layer Perception, K-Nearest Neighbour, Random Forest, Logistic regression, and XGBoost in layers for effective pattern detection with less false positive rate. The model is robust against any evolving threat, however demanding high computation and keeping in consideration of hyperparameter tuning.

G.S. Nayak et al.[2] presents a detection system for phishing based on the fusion of feature selection and deep models. The technique enhances the accuracy of the detection by using appropriate features. In this respect, it combines machine learning algorithms to achieve efficient classification. It equally demonstrates its competence in adapting to a sufficiently dynamic environment with respect to phishing threats.

W. Guo et al. [3] have developed a novel phishing URL detection method using loopy belief propagation and graph-based machine learning. The novelty of the work lies that the authors have transformed URLs into graphs capturing structural relationships among features forming a basis for improved efficiency for false positives and enhanced detection accuracy. The method is however computationally expensive but shows a remarkable potential towards scalable real-time phishing detection. This has proved especially effective against intricate phishing patterns.

M. A. Daniel et al. [4] conducted a comparison of several Machine Learning methods in Phishing Detection with feature selection. This research makes it necessary to gather a certain number of the optimal detection performance features. It demonstrates model performance improvement, complexity reduction, and evidence-based support for selecting an appropriate algorithm among different ML techniques for using in phishing defense. It also highlights the trade-off between efficiency and effectiveness for use in detection systems.

Ozgur Koray Sahingoz et al. [5], have developed a phishing detection system called DEPHIDES. The system employs deep learning models designated as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Bi-directional Recurrent Network (BRNN). Despite the high computational and frequent update demands, the advantages include accuracy and robustness for deep feature extraction.

Remya et al. [6] proposed a phishing URL detection model using Residual Multi-Layer Perceptron (ResMLP) which exploits both the structured and unstructured features of URLs for improved accuracy. But the model has few limitations such as overfitting and a fairly high demand for computational resources. Future work is intended to focus on enhancing feature selection and utilizing adaptive learning to counter the changing trends of phishing attacks.

Ahmad Sahban Rafsanjani, et al. [7] introduced an entirely new framework for detecting malicious URL detection engine based on adaptive feature ranking and machine learning, which improved performance with a reduced false-alarm rate. Few disadvantages of the proposed framework are the scaling and real-time operations and continuous heavy updating required for the framework to detect new phishing patterns.

W. Li et al. [8] conducted a systematic review of techniques used to obscure phishing websites. The study discusses different cloaking techniques that the fraudsters use to avoid detection and assesses the efficiency of existing detection mechanisms against those obfuscation schemes. Major contributions include taxonomy of cloaking strategies and suggestions for countermeasures. Thus, the review might also prove useful in delivering stronger solutions to combat phishing attacks.

D.J. Dsouza et al.[9] conducted a comparative study of the AI technology based multi-modal analysis for phishing detection. The study evaluates several AI models across different kinds of data and their effectiveness. The emphasis of the study is on accuracy, adaptability, and performance of the models, in choosing the best AI model to combat against phishing attacks.

Almousa et al. [10] proposed adopting a URL-based phishing detection system that deploys character-aware language model to detect garbled URLs during social engineering attacks. The system achieves great accuracy in its day-to-day prevention, requiring high costs for its training and also struggling with advanced obfuscations. However, it does increase phishing detection using deep learning with text analysis capabilities.

Abdul Karim Samir et. Al [11] proposed a hybrid phishing detection system for better accuracy combining decision trees, support vector machines, and deep learning. Lexical, host-based, and network-based features are applied for effective classification of URLs. Therefore, the proposed model achieves better detection efficiency over traditional single-model approaches. The disadvantages of the proposed system are the feature selection, cost for computation and adapating capability. Continuous improvement needs to be done to keep up with the changing phishing threats.

Alwyn R. Pais et al. [12] discussed a phishing detection system using boosting-based hybrid feature selection along with a multilayer stacked ensemble learning model (MLSELM). The combination of XGBoost, Random Forest and logistic regression helps to boost the accuracy. Highlights of the research include the proposed model's utilization of both relevant feature selection and diversity of models. The complexities and adaptability of the model are some of the disadvantages.

Belhaouari et al. [13] presents a novel approach based on machine learning towards the detection of phishing URLs. This approach is an efficient means for detecting phishing URLs and scores an accuracy of 97.85 percent with regard to detection, precision, and recall. The paper emphasizes adapting to new phishing methods over time and plans to strengthen feature selection on-the-fly around better cybersecurity measures.

Ahmet Ozaday et al. [14] have performed URL classification on six machine learning algorithms along with the 11 features. Among those six ML algorithms, Random Forest was found to have the highest accuracy as 98.90%. The researchers have come to the conclusion that Random Forest has always been more effective and consistent among the other ML models for phishing detection.

R. Zieni et al. [15] have conducted an overview survey into phishing website detection. Their work concerns the description of a number of techniques for phishing site identification and effectiveness evaluation. The techniques from machine learning models to heuristics were discussed in the survey. The challenges and future directions for phishing detection systems lies on future implementation of phishing detection systems. Thus, the survey serves as an overview of the entire field of phishing website detection.

Manuel Sánchez-Paniagua et al. [16] describes the real-world data taken in machine learning to increase accuracy rates for the phishing URL detection system suiting login URLs structurally and lexically. However, the drawbacks include obfuscation of URLs, zero-day phishing attacks, and evasion tactics. Suggested future work includes improving feature engineering and tying in real-time threat intelligence to improve detection capacity.

T. Mummadi et al. [17] conducted a survey that depends on a supervised learning algorithm. The key focus is developing algorithms for practical application such as decision trees and the SVM to classify highly accurate cases. This paper emphasizes working between the methodologies of research and detection of phishing sites. The study proposed features extraction of URLs for phishing sites really efficient and very much appreciable for real-time detection purposes.

Ilker Kara et al. [18] presented a phishing detection system based on machine learning for analysis of URL and domain features. It employs lexical, domain-related and statistical attributes, which are processed by decision trees and ensembles to provide an accurate model. The system detects phishing attempts effectively however the feature selection, evolving attacks, and datasets are some of the challenges. It promotes the effectiveness of URL and domain attributes for strong detection in this work.

Maria Sameen et al. [19] developed a real-time AI-based phishing detection system named PhishHaven. The system offers analysis of URL structures and web page features. There is support for browser and an adaptive detection mechanism. However, challenges include high computational demands and the organization of labelled data.

S. Bell et al. [20] analyzed the effectiveness of distinguished phishing blacklists from open source phishing site such as Google Safe Browsing, OpenPhish, and PhishTank. The study compares coverage, update frequency, and accuracy of detection. It discusses inconsistencies and delays in updating blacklists. The results will help in better utilization of blacklist-based phishing defenses.

CHAPTER 3

PROBLEM DEFINITION AND METHODOLOGIES

3.1 PROBLEM DEFINITION

As dependence on online services and digital platforms increases, the risk posed by a phishing attack has gone up significantly. Cybercriminals trick their users through fake websites by using a misleading URL and asking the users to provide sensitive information such as passwords, credit card information, and personal details. Such increasing complexity of phishing makes it more difficult for users to tell the different between the sites and therefore poses a high threat to information security and privacy.

Challenges remain in detecting such phishing websites due to the dynamic nature of their URLs and constant changes that evade detection. Any traditional detection methods like blacklist-based systems and rule-based filters, however, may not be effective for this type of newly emerging phishing attacks. Recognition of such websites relies on pre-defined information, hence such methods would fail to identify the so-called zero-day phishing attack. Thus, this project includes such issues and suggests an automatic phishing detection system, which using machine-learning techniques classifies URLs into phishing or legitimate ones, thereby increasing accuracy in detection and protecting users from online hazards.

3.2 EXISTING SYSTEM

Traditional phishing detection methods often depend on blacklist-based systems and rule-based techniques to identify malicious websites. However, these methods struggle to detect newly created phishing websites that haven't been reported yet. While these methods offer some level of protection, they possess several critical limitations.

The different traditional ways of detecting phishing have some inherent limitations that affect their successful implementation such as detection of zero-day attacks or emerging phishing techniques. The blacklist method deals mainly with already existing URLs that have been tagged as phishing and thus cannot protect against zero-day phishing attacks because the URLs are brand

new. Similarly, static rule-based detection systems have fixed patterns for classification yet will never change these patterns to counteract the ever-changing scams of criminals. Generally characterized by increased false negative rate especially in the cases of obfuscated and shortened URLs leading to inappropriate classification issues.

Furthermore, most of the existing systems lack feature extraction and only consider surface-level URL characteristics ignoring hidden patterns that could be embedded into URL structures. Moreover, the described approaches of existing system relying on traditional black-listing or static rule based techniques fail to provide detection of phishing attacks in real-time and generation of instant alert.

Disadvantages of the Existing System:

- Traditional phishing detection models have difficulty in coping with the innovations of phishing techniques being brought into practice today, thus affecting their accuracy to an extent and raising their false-positive rates.
- High computation needs limit the deployment to resource-constrained environments making it a barrier to real-time performance.
- Existing solutions lack adaptive learning mechanisms to counter the emerging attacking techniques against phishing.

3.3 PROPOSED SYSTEM

An effective phishing detection system is created to utilize real-time URL analysis integrated with machine learning approaches to advanced cybersecurity. The proposed system employs the MLSELM and XGBoost for improved accuracy detection of phishing websites. The objective is to prevent phishing attacks through URL analysis and alerting users before they visit a possibly infected site.

The system is designed as a chrome extension which provides a seamless user experience and monitors URLs. However, it is enhanced with advanced feature selection techniques like TD-IDF score because it helps in detection models to distinguish between legitimate and phishing URLs.

The project's applications range from online banking and corporate security with improved personal browsing. The system increases awareness of phishing and means of protecting users from it. The system architecture offers scalability hence additional large datasets and complicated phishing attack patterns could later be integrated for greater effectiveness. By a mixture of AI-driven detection with real-time monitoring, this project addresses a very efficient and adaptive approach along the defense against phishing and ensures secure browsing experience for the user.

Advantages of the proposed system:

- Direct real-time process of analyzing URL for detection and blocking phishing attempts before a user even interacts with it.
- These include some of the advanced machine learning models such as Random Forest and XGBoost improving accuracy and reducing the false positives in phishing detection.
- Conveniently deployed as a Chrome Extension, designed to ensure seamless and user-targeted protection during the various types of browsing activity.

Constraints of the proposed system:

- The performance of the system may suffer from temporary slowdowns due to the real-time URL analysis, which may affect the browsing speed and user experience.
- Accuracy of URL classification relies on the input training datasets; however, dataset maintenance for accuracy poses a challenge.
- The system should be continuously updated to accommodate new phishing attacks.

3.4 METHODOLOGIES

This system comprises the following major phases that are dataset preparation and preprocessing, feature extraction, machine learning model development, model training and evaluation, extension development, and backend server integration. Dataset preparation and preprocessing will involve downloading the phishing and legitimate URL datasets from sites such as Kaggle or UC Irvine Machine Learning Repository, cleaning, and normalizing the data

thereafter. Feature extraction will involve extraction of relevant URL attributes by characterization, such as lexical, host-based, and content-based features, combined with TF-IDF for transformation. Machine learning algorithms such as XGBoost, and Random Forest are to be trained with the extracted features to detect malicious URLs. Model training and evaluation deals with the fine-tuning of models using hyperparameter optimization and evaluating them against accuracy, precision, recall, and F1 score. The URL monitor, which gives instant feedback to the users on possible phishing threats is enabled in real time as chrome extension development. Lastly, backend server integration ensures dynamic analysis of phishing URLs thereby keeping the phishing system to continually adapt to new threats.

3.4.1 Dataset Collection and Preprocessing

In this section, collection and preprocessing of datasets related to a phishing detection system will be discussed in terms of how different types of URL data are used for analysis. Through cleaning, normalizing, and structuring features, the data is transformed for better optimization of model training and phishing identification.

Dataset Collection:

Firstly, the phishing detection compiles the dataset of phishing and legitimate URLs initially sourced from Kaggle and UC Irvine Machine Learning Repository. The dataset consists of different categories of URLs covering a multitude of phishing techniques and legitimate structures.

Preprocessing:

All the URLs collected in the process undergo preprocessing. This facilitates data cleaning and duplicate removal such as special characters, URL encodings and unnecessary parameters, while assuming all these changes are consistent without affecting the dataset. Missing values and irrelevant attributes are then removed to increase model performance.

Feature Extraction:

Feature Extraction is used to distinguish between phishing and legitimate URLs. Basic URL features such as lexical features-which include length of the URL and symbols-used, host features which comprise domain age and

WHOIS information, and content features, which include embedded JavaScript and redirection-methods. These features are extracted for further training of the model and phishing detection.

3.4.2 Feature Extraction

This segment emphasis on the critical feature extraction process from lexical, host-based, and network-based attributes along with an analysis of URL patterns based on TF-IDF. These features are used for model training, so they undergo feature selection techniques that analyze them to retain only relevant features, optimizing the efficiency and performance of the model.

Lexical Features:

Lexical features concentrate more on the internal structure of a particular URL, for example URL lengths, the occurrence of special characters in the URL, the number of subdomains, and suspicious keywords such as login, verify, and secure. All these are helpful to classify the phishing URLs to real and legitimate websites.

Host-Based Features:

Domains include WHOIS, age of the domain, an IP-based host, and other new or strange domains that constitute the additional class of host-based features. This makes most phishing websites use these short-lived domains and manipulate it to create phishing URLs.

Network-Based Features:

While describing network-based features, these involve servers and their aspects such SSL certificates, DNS records, redirection behavior, and response time. Most phishing sites self-sign their SSL certificates or have invalid ones and exhibit strange redirecting behaviour.

TF-IDF for URL Analysis:

TF-IDF means Term Frequency-Inverse Document Frequency. It determines the resource of meaningful text patterns from URLs. This is used for the classification of URLs as either legitimate or phishing by means of a sequence analysis of character and keyword distributions.

Machine Learning-Based Feature Selection:

Features selection methods based on statistical analysis and computation can be proved effiicient in improving the model performance. Such features are applicable in enhancing the efficacy of the. Therefore, it will keep the computation overload reduced and produced more accuracy in detection of phishing website.

3.4.3 Machine Learning Model Development

This section describes the phases of machine learning model such as feature engineering, model selection, and ensemble model construction. The developed model is used for evaluating, optimizing, and refining the framework of the MLSELM ensemble for phishing detection. The proposed ensemble model provides increased phishing detection through better performance in accuracy and detection.

Feature Selection and Engineering:

The lexically extracted host-specific and network-specific features statistically tested for similarity selection would optimize a model for both robustness and performance running time. The model eliminates features that are unimportant or not essential because it improves the overall performance.

Model Selection:

The comparisons on various machine learning algorithms show that the chosen algorithms have been selected due to their performance. XGBoost and Random Forest models have more suitability due to their accuracy and robustness while observing complex patterns. Hence, they were chosen for the ensemble model.

Adaptive Learning with MLSELM:

The MLSELM works by improving the accuracy rates in phishing detection by stacking of XGBoost and Random Forest. The model adopts such an adaptive learning methodology in which it chooses the majority voting among the ensemble models in order to generalize itself against the changing adaptive techniques of phishing.

Model Optimization and Hyperparameter Tuning:

The usage of grid search and cross-validation improves detection accuracy and decreases false positives when processing XGBoost and Random Forest hyperparameters. Feature importance analysis functions to enhance the model performance.

Training and Evaluation:

The final ensemble model is trained using labelled phishing and legitimate URL datasets. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to measure the model's effectiveness in real-world phishing detection scenarios.

3.4.4 Model Training and Evaluation

In this section, the phishing phases are detected by MLSELM-based models, which are involved in training and evaluation concerning data set arrangement, hyperparameter tuning, and performance evaluation clauses. These highly beneficial set of evaluation parameters and further feature analyses help refine the model to the better accuracy and reliable recognition of phishing threats.

Training the Model:

The implemented MLSELM recieves the extracted features to train the ensemble model which consists of XGBoost and Random Forest. The various features extracted from the dataset are randomly divided into their training and testing sections, with an equal representation of phishing and legitimate URLs.

Hyperparameter Tuning:

Hyperparameter tuning is used for grid search and cross-validation to achieve improved model performance. Some parameters that have been tuned include learning rate, tree depth, and the number of estimators for XGBoost and Random Forest which results in improvement of accuracy and eliminates overfitting.

Evaluation Metrics:

The evaluation is done on the various performances obtained from these metrics to determine whether or not the obtained trained models actually work when it comes to detecting phishing URLs. The performance metrics are the accuracy metric would measure correctness against the number of URLs that have been properly classified in reference to the total number of samples. Precision would define how many of the URLs classified as phishing would actually be phishing, thus reducing false positive findings. The model should be checked on the recall side on how much it could find phishing URLs from real threats that they may have missed. The F1-score is an overall measure that shows how the model performs on precision and recall. The confusion matrix also provides an evaluative framework to study how the performance of the model is doing with respect to the distribution of false positives and false negatives and further looking into possible weaknesses. With these evaluation metrics, the phishing detection system ranks better in accurateness, efficiency, and ability to differentiate between legit URLs and malicious ones.

Performance Analysis:

Grid search and cross-validation contributes hyperparameter tuning, which increases the performance of the model with respect to accuracy. The learning rates, tree depths, and number of estimators for XGBoost and Random Forest have been tuned improving the accuracy to reduce overfitting.

3.4.5 Chrome Extension Development

This section discusses about the implementation of the Chrome extension for phishing detection. The extension will provide the user with project output such that when the user types any URL in the chrome extension input, it will display below whether it is safe or phishing.

Extension Architecture:

The chrome extension is being integrated by the web browsers to facilitate real-time monitoring of URLs for phishing attempts. It is implemented using a background script, content script, popup interface, and messaging system.

User Interface Design:

With hyper-text markup language (HTML), cascading style sheets (CSS), and JavaScript, a simple popup interface is created for user convenience. This immediately indicates whether a URL is safe or malicious and returns the result as legitimate or malicious.

Real-Time URL Monitoring:

The extension captures URLs from URL bar and submits them to the MLSELM phishing detection model to conduct interviews on them. The usage events of a user are monitored using webRequest API (onBeforeNavigate), making it possible for some suspicious URLs to be checked before a user proceeds to the website.

Integration with MLSELM Model:

This extension captures URLs from the browser tab and submits them to the MLSELM phishing detection model to predict the possibility of phishing website. The webRequest API tracks the URL in real-time as the user types making it possible for some suspicious URLs to be evaluated before visiting the webpage.

Deployment and Testing:

After the integration of flask server within the chrome extension, the model is deployed for real time phishing detection. It is tested using various safe and malicious URLs to determine the accuracy of correct and false detections. With the received output, the model is further trained to improve accuracy if it results in more false positives. If it is correctly detecting, it is continuously updates to the emerging manipulations made in URL phishing.

3.4.6 Backend Server Integration

The system achieves phishing detection through optimization which decreases API call overhead and improves URL processing speed. The system uses caching approaches to reduce system load from unnecessary calculations which in turn enhances response speed.

Server Architecture:

A Web hosting service set up for that local server provides phishing detection

APIs enabling exchanges between the user interface and back-end systems. This API has a code written under Flask which provides efficient lightweight processing requests. The data of the whole interaction of the users is secured with SSL/TLS that makes use of very secure encryption protocols.

API Development:

The REST API was built as a bridge for communication between the Chrome extension and the backend. The API receives URLs sent from the extension, extracts features from those URLs, processes them using the MLSELM model, and returns a phishing classification value (Safe, Suspicious, or Phishing).

Feature Extraction & Model Processing:

The server receives URL, then it extracts some key features from the URL such as domain age, length of the URL, the presence of special characters in the URL, whether the URL has HTTPS, text features that are vectorized using TF-IDF and these features are given to the MLSELM ensemble model to predict whether the URL is legitimate or fraudulent.

Deployment & Scaling:

Backend is hosted either on cloud server or local set up with high availability and scalability for running the application. Load balancing and caching are optimizers that allow serving multiple demands concurrently. Periodic releases are done to improve the phishing detection accuracy and to extend the threat intelligence sources pool.

3.4.7 Phishing Detection Deployment

This segment discusses about the deployment of a phishing detection system for the identification of threats in a secure, real-time manner.

Deployment Environment:

The phishing detection service operates from an on-premises server that possesses dedicated resources for dependable management of its deployment environment. The MLSELM-based phishing detection model executes on the backend server while the system functions without depending on third-party cloud-based services.

Web Server & API Hosting:

The local server is set up with a web hosting service that provides phishing detection. The APIs are used for communication and has a code written under Flask which provides efficient lightweight processing requests. The API ensure that the phishing detection process between the Chrome extension and flask server is smooth without delays.

Model Deployment:

The MLSELM model runs locally or in a cloud environment. Real-time URL request processing is enabled by the system to identify phishing threats accurately. The model is updated manually for improved detection performance.

Performance Optimization:

The system achieves phishing detection through optimization which provides better accuracy and detection. The system also ensures elimination of overfitting dataset. It also ensures the smooth communication handling of API between flask server and chrome extension.

CHAPTER 4

DESIGN PROCESS

4.1 DESIGN OVERVIEW

Multiple components of the design system operate through a modular architecture which achieves effective phishing detection. The system arranges its components into separate modules which handle both feature extraction tasks along with machine learning model development along with Chrome extension integration and backend server deployment functions for real-time phishing detection.

The process is carried out by the TF-IDF techniques transform URL characteristics into numerical values for feature extraction purposes. The phishing detection model receives extracted features from its input layer before employing MLSELM for its classification process.

In real-time the Chrome extension module processes the URL typed by the user and analyzes it then exchanges data with the backend system for phishing threat determination. The backend server receives user requests after which it implements model execution to assess phishing risks which the Chrome extension eventually receives for display.

The integration component between modules maintains efficient data exchange which enables complete phishing threat detection as well as response workflow. Every project cycle starting from dataset acquisition through preprocessing steps continues to model training before evaluation then reaches system integration to deployment phases as the solution operates effectively while being scalable.

Continuous modifications are performed to the model by assessing evaluation metrics and implementing feedback data from testing. This design delivers a reliable phishing detection solution which builds web security to protect users from dangerous websites.

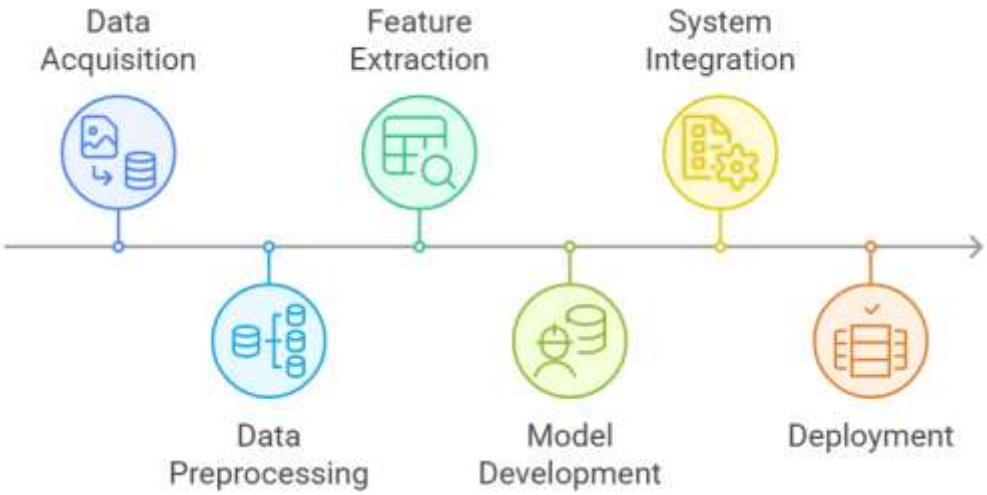

Fig 4.1 Design process for Phishing Detection System

Design process steps:

The projects are divided into six stages. The first stage is data acquisition where phishing and legitimate URLs are collected from sources like Kaggle and UCI. Data preprocessing is the second stage where the URLs are cleaned and then the formats are checked to have enough consistency. In feature extraction, URLs are converted into numerical vectors via TF-IDF. In the model development section, algorithms like Random Forest, XGBoost are merged into the MLSELM which are then trained and tested. System integration puts the model into a Chrome extension for running detection in real time. Finally, deployment is where the host of the system using Flask for detection of phishing websites and chrome extension is used to display and alert the users about potential phishing attacks.

- **Data Acquisition**

  The dataset preparation begins with phishing and legitimate URLs sourced from phishing datasets in order to create an extensive collection. The URLs will be labelled according to their nature as phishing or safe for supervised learning purposes. Text URL data will be machine-readable through feature extraction by means of TF-IDF vectorization, thus converting the text data into a numerical representation to apply in effective model training.

- **Data Preprocessing**

  URL collection requires data pre-processing that converts the harvested URLs for readiness after cleaning them. The next step involves TF-IDF vectorization that derives useful textual elements from URLs. The detection of special characters and numbers and alphabets using character frequency-based feature extraction helps produce more diverse attributes. The model accuracy strengthens alongside the elimination of irrelevant data and completion of missing values to maintain dataset consistency for training purposes.

- **Model Development**

  After splitting the dataset into the training and testing sets, the model training process adopts the train-test split process. After this, it extracts these features to train Random Forest (RF) and XGBoost independently. These two models will then use the training dataset to learn about phishing patterns that will help them recognize suspicious URLs and improve detection performance.

- **System Integration**

  As a monitoring interface, the Chrome extension captures the URL in real-time while the users type in the input box. The URLs are sent through HTTP requests to the Flask API, where in the backend processes them and checks if they are phishing attempts. The result of this checking is sent to the users in the chrome extension itself.

- **Deployment**

  The Flask API operates from a self-local server which receives all incoming URL requests. The backend process includes using Random Forest and XGBoost ensemble models for real-time phishing attack detection in collaboration with the Chrome extension development and deployment. The system has implemented a URL classification feature which operates during user browsing to deliver immediate output without any dependence on external monitoring services.

4.2 DATA FLOW DIAGRAM

The flow of data processing through various sequential steps is illustrated in Fig. 4.2, starting with dataset collection through gathering URLs from well-trusted repositories such as Kaggle and UCI Machine Learning Repository. The URLs are subjected to preprocessing and further categorized as either phishing or legitimate. A variety of URL-based features are automatically taken as part of the feature extraction stage for domain structure analysis, special character detection, and protocol examination.

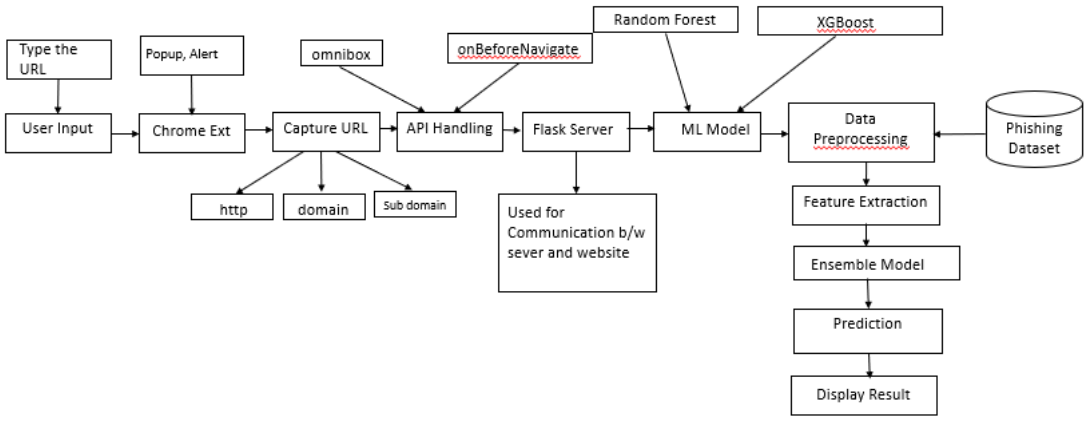

Fig 4.2 Data Flow Diagram for Phishing Detection

The machine learning model development stage then takes the extracted features from preprocessing to allow separate training of Random Forest and XGBoost algorithms. For gaining a distinctive capability in separating phishing URLs from normal URLs, these two models make a concerted effort to learn the distinguishing patterns associated with the extracted features. After the train-test split procedure. The analysis makes use of accuracy, together with precision and recall, using the F1-score as a model performance metric. Then the models are combined to form an ensemble model.

The ensemble model merge the Random Forest and XGBoost prediction outputs through a Voting Classifier that enhances detection accuracy while lowering false results. The resultant trained ensemble model is then uploaded to the Flask backend server for implementation.

The Chrome Extension detects a malicious URL while streaming URLs to the Flask backend via HTTP requests. The server platform receives the requests made by the user and processes them through the ensemble model trained on

the URL and responds back to the user. The Chrome extension then presents the acquired result for real-time display to the user via the chrome extension.

The system is integrated with the chrome browser to detect the phishing websites using the data flow system that provides the entire flow. This helps in understanding the working of the phishing detection model and how it predicts whether a given URL is phishing or legitimate.

4.3 ARCHITECTURE DIAGRAM

The Chrome extension collects URLs browsed on the browser and sends them to backend servers for analysis. The Flask API has some machine learning models trained on phishing datasets for processing input in terms of analyzing URL characteristics extracted through TF-IDF and pattern detection engine for identifying legitimate or phishing sites. The extension uses the backend result for enabling an alert on the detection outcome. If the extension finds a phishing site, it notifies in the chrome extension preventing lead of sensitive information.

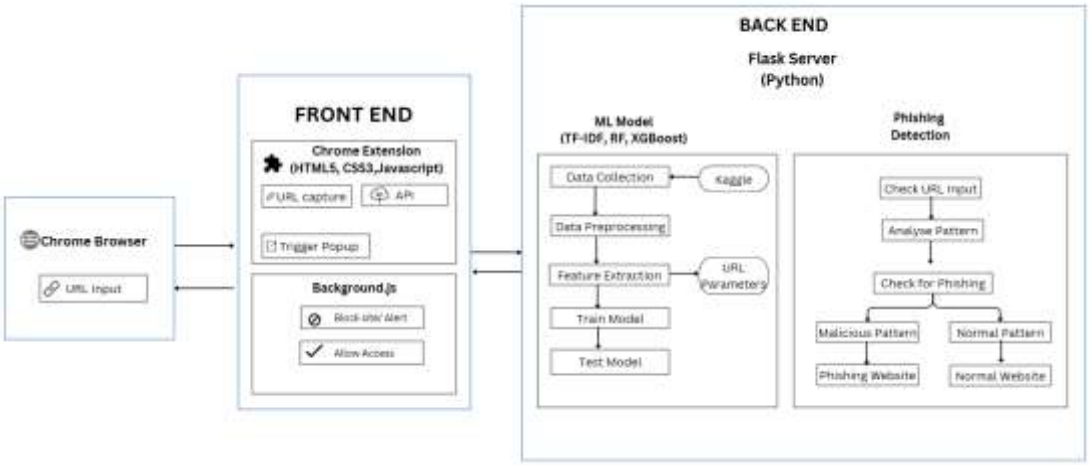

Fig 4.3 Architecture Diagram for Phishing Detection

The above diagrammatic representation in Fig 4.3 clearly shows how data is transferred from one end to the other. The programmed data sequence, which is shown from URL collection to phishing identification until alerting the users, results in a real-time prevention of phishing attacks, which increases the chances of protection while offering seamless browsing sessions. The system makes use of both the frontend browsing abilities and backend machine-learning enabling immediate phishing alerts while managing the false-positive

phishing warnings. The system ensures smooth user experience and safe guards against phishing attacks in an effective way.

4.3.1 User Interface

Users can check the safety of websites in real time using the extension integrated into Google Chrome browser. A URL entered by the user will automatically go to the backend Flask server for analysis. The user interface is designed using HTML, CSS3, and Javascript to ensure seamless and user-friendliness of the extension. Once the detection is complete, the output is shown in an extension indicating whether the given URL is a safe site or potentially a phishing site. Background scripts check the URLs and interface, enabling real-time detection of phishing websites in order to improve usability and safety.

4.3.2 Front-End

HTML5, CSS3, and JavaScript comprise the system's front end, which are the major parts of the interface for the Chrome extension. The interface is simple, clean, and responsive so that users can easily type the URL in the browsing environment. After the URL is pasted into the extension, API calls are made to the back-end server. The URL alert is displayed in the chrome extension, showing the results in real time alerting users if the site is phishing or safe. The background scripts take care of permissions, alerting any malicious sites or allowing them for a safe browsing experience, thus providing smooth interaction.

4.3.3 Back-End

During this stage, machine learning models such as those of the Random forest and XGBoost algorithms will be designed to classify URLs in a way that determines if they are either phishing or legitimate. The features extracted from the dataset are sent to these models which allow them to learn patterns or points of interest typically in phishing attacks. At the same time some hyperparameter tuning and cross-validation optimization has been performed so that the models developed in this phase are at a good

level of accuracy and precision while maintaining low rates of false positives. These trained models will later be built into the detection system for real-time classification application.

4.4 PROJECT REQUIREMENTS

To effectively carry out this project, specific technologies and tools are required. These include both software and hardware components that support the development, training, testing, and deployment phases.

PYTHON

Python serves as an adaptable programming language which implements object-oriented as well as procedural methodology. The programmer Guido van Rossum created this programming language which provides an elegant syntax alongside simplicity thus becoming suitable for novice developers and professionals. The language operates without limitation on various operating systems through its independent platform feature. The widespread popularity of Python stems from its universal functionality which includes graphical user interface application development.

Some of the key features of Python include:

1. Simple and Easy to Learn
2. Free and Open-Source
3. Supports Graphical User Interface (GUI) Programming
4. High-Level Programming Language
5. Integrated with Other Languages
6. Interpreted Language

Libraries used:

Various libraries have been used for this project which are listed below. These libraries are used for machine learning and server hosting.

Flask

Flask consists of a versatile web framework for Python which provides developers simple means for rapid web application development. Web

programmer Armin Ronacher designed Flask with a minimalistic design which allows developers free selection of their preferred tools and libraries. Flask includes three primary components: built-in development server functionality as well as Jinja2 template capability and URL routing features. While simple in design Flask provides a wide range of capability needed for advanced web applications and proves popular among professionals and novices in web application development.

Sci-kit learn

The open-source machine learning library named Scikit-learn delivers basic and efficient tools for conducting data mining along with data analysis using Python. Scikit-learn functions as an open-source Python library which extends NumPy, SciPy, and matplotlib offering many classification, regression, clustering and dimensionality reduction and model selection algorithms. The consistent API structure makes scikit-learn popular among industrial and academic researchers for developing robust machine learning models easily.

CHROME EXTENSION

Chrome Extension functions as a compact program that increases the performance capabilities of Google Chrome browser. Web technologies including HTML and CSS together with JavaScript enable Chrome Extensions to add new functions which help users manage ads and passwords and other productivity tools inside the Google Chrome browser. Chrome Extensions utilize web technology to participate with web pages and browser APIs so users obtain better integration together with enhanced interaction capabilities.

Some of the key features of Chrome Extensions include:

- Easy Installation and Usage
- Customizable User Interface
- Access to Chrome APIs
- Background Script Support
- Lightweight and Fast
- Secure and Sandboxed Environment

SOFTWARE REQUIREMENTS

Software requirements are basically the necessary for a system, telling the user what and how things should be implemented. It's crucial for designing, developing, testing, and keeping the software running smoothly.

Operating System: Windows

Tools: The system demands two components for development and execution: Python as programming language and Scikit-learn alongside other machine learning libraries for model building. The Pandas along with NumPy handles data processing tasks while Flask serves as the framework for designing the minimal yet powerful backend application programming interface. The development along with testing of the complete project is executed inside the Python IDE which delivered both good efficiency and a smooth user experience throughout implementation and debugging phases.

HARDWARE REQUIREMENTS

Processor: A computer system needs a processor equipped with multiple cores along with robust speed capabilities for efficient operation of machine learning operations. A system performance of Intel Core i5 or above provides sufficient power for both training models and active URL analysis tasks.

Hard Disk: The system needs a sufficient amount of storage from the hard disk for data management activities including dataset storage together with model retention and extension file management. The preferred SSD for the application should have at least 256GB capacity because it provides both quick read/write speeds and optimal performance.

RAM: At least 8GB of RAM is required to support smooth execution of the phishing detection models and browser extension integration during development and deployment.

CHAPTER 5

IMPLEMENTATION

By uniting machine learning phishing detection methods with URL monitoring in a Chrome extension the system gives users one solution to detect and block harmful websites. The system achieves real-time phishing alerts by performing URL analysis through a merger of feature extraction and ensemble models at the browser level. The complete system design strengthens user protection for online banking together with e-commerce and secure browsing because it delivers immediate defense against phishing attacks.

5.1 PHISHING DETECTION

Deceptive methods used by phishing attacks have enabled significant evolution during the recent years to extract sensitive information from users. The existing detection systems struggle to identify phishing URLs because these URLs present an evasive dynamic nature. The project introduces machine learning capability which learns multiple URL features from various datasets to detect evolving threats in phishing patterns.

The project starts with the phase of gathering and preparing phishing and legitimate URL datasets. A feature extraction process derives significant characteristics from URLs such as their length together with aspects on special characters domain age metrics and keyword patterns. These elements serve a vital function during the machine learning process because they help models recognize diverse URL characteristics associated with malicious and safe URLs.

This project executes phishing detection with ensemble machine learning models Random Forest and XGBoost that function together to classify URLs through the extracted features. Machine learning models receive training by working with datasets which combine both phishing and legitimate URLs which enables them to master the one-of-a-kind patterns of malicious links. The detection methodology maintains both precise accuracy and efficient speed of detection.

The ensemble method achieves more reliable classification results by letting models vote on predictions therefore decreasing the number of wrong positive detections. An effective URL distinction between safe websites and harmful websites becomes possible through system analysis of structural and lexical URL attributes. Such combination strengthens the model's protective capabilities against current and emerging forms of phishing attacks.

The ensemble model receives integration to a Flask server backend for URL monitoring through a Chrome extension that performs real-time execution. The extension retrieves browsing URLs which are immediately sent to the backend framework for analysis. The system provides the output from the flask server and determines whether a given URL is legitimate or phishing helping to enhance security.

5.1.1 Data Sourcing

An effective intelligent system should rely on data collection as it is major for the entire phishing detection project. Some trusted data sources such as Kaggle and the UC Irvine Machine Learning Repository contains datasets with multiple phishing URLs and legitimate URLs. The datasets contain both malicious and safe links used as real-world examples in training data that positively enhance the accuracy and performance of the system.

URL entries include attributes having URL lengths, a special character status, HTTPS site usage, domain age details, and redirection behavior. The model makes use of these attributes to capture those difficult characteristics that separate phishing domains from regular internet addresses. A2.1 and A2.2 shows the datasets which consists of various number of attributes of the URL checked against various websites to determine phishing or safe.

The URLs collected need cleaning from unnecessary duplication of records, managing missing values, and adjusting the formatting of data in a more normalized manner before proceeding with the training process. Cleaning ensures the dataset becomes consistent as it proceeds since consistent data make for better environments in feature extraction and model training. The progress made is important for the future development of the model because it makes up the primary component to undergo this step.

The cleaned dataset that is well balanced increases the chances of robustness of the detection system because it will be able to perform well over many phishing methods to which the system may be subjected. Hence, accurate predictions are generated, and false alarms reduced while improving user security in real time for phishing detection purposes.

5.1.2 Feature Vectorization

The fundamental role within the phishing detection system belongs to feature extraction because it converts unstructured URLs into machine learning data. The system extracts features from each URL through parsing before it recognizes the standard features that indicate phishing activities. Four core technical features organizations use for phishing detection include checking for IP locations in domain names as well as inspecting special characters "@", evaluating domain length restrictions and analyzing deceptive subdomains.

The features from phishing URL extraction are chosen specifically because they demonstrate strong capability in detecting legitimate vs phishing targets. The typical features of phishing URLs consist of excessive hyphens together with the incorrect usage of HTTP rather than HTTPS and deviation from genuine domain names. The features undergo systematic calculation leading to numerical or categorized structures which machine learning algorithms can comprehend.

The extracted features are put into a data collection structure that uses URLs as rows while features perform as columns. The defined input structure allows effective training of reliable models including Random Forest and XGBoost since these algorithms require structured input to learn predictive phishing patterns.

The overall success of the phishing detection system stems from extracting features of high quality and proper relevance. Selecting the right features delivers improved accuracy in classifying URLs and promotes the system's ability to handle previously unseen URLs. The implemented feature engineering methods generate a reliable outcome for real-time phishing threat detection.

5.2 TRAINING THE PHISHING MODEL

The core aspect of creating a phishing detection model involves its training to distinguish genuine URLs from phishing URLs through previously acquired features. The base data needs to be split into training and testing segments for accurate evaluation purposes with an appropriate size ratio. Model training along this method allows particular sample data to build accuracy while new data ensures proper validation for preventing overfitting problems.

The model used two approaches for URL feature extraction: separate training by random forests followed by xgboost. Random forest was selected because it works with complex data types and xgboost functions rapidly with large quantities of data. New components involving detection of fraudulent websites against original ones continue to evolve within these two developing models.

The training parameters go through an optimization process of grid search together with cross-validation methods which allows performance evaluation to determine the most suitable parameter configuration for accuracy elevation and overfitting mitigation. The models recognize URL features as core phishing elements because they allow them to build specific decision boundaries.

The testing of trained models requires accuracy measurements along with precision, recall calculation and F1-score evaluations for thorough evaluation. The model's reliability together with its ability to handle new data constitutes its main evaluation metrics. The practical model testing uses confusion matrixes to demonstrate actual performance yet it simultaneously exhibits both positive and negative aspects.

The prediction output of the trained model can be found in Windows Poweshell using Invoke-RestMethod within Fig A2.3 - Fig A2.6. The prediction output by trained models determines whether a URL belongs to phishing or legitimate classification.

5.2.1 Random Forest

In this project, the principal detection system for the recognition of phishing URLs is the Random Forest machine learning classifier. Random Forest is an ensemble-matrix approach that trains multiple decision trees during training. The methods used by the trees are combined to obtain better classification accuracy and control for overfitting. Random Forest is considered for phishing detection because it provides good data robustness in large-scale dataset handling and efficient processing of high-dimensional data.

In implementation, the algorithm randomly selects subsets of both training data and features so that multiple decision trees can be constructed. The final prediction of each tree classifies URLs to either legitimate or phishing by checking features such as URL length and domain age, along with other features. The model then puts all individual trees output predictions back together using a voting system to generate a final prediction. The method enhances prediction generalization and minimizes errors that arise from noisy data while reducing overfitting risk.

The training process of Random Forest uses a set of data containing legitimate and fraudulent URLs for model development. The trained model gives one base learner to the ensemble learning setup, which works with the XGBoost to give an output in the Voting Classifier. The ensemble improves phishing detection accuracy in real time by providing trust over the resulting system.

Algorithm:

For i = 1 to N do:
Sample dataset Di from D using bootstrap sampling
Build a decision tree Ti:
For each node in Ti:
Randomly select m features from total features
Determine the best split using chosen features (e.g., Gini Index)
Split the node into child nodes
Repeat recursively until stopping criteria met
Add Ti to RF

End For

Prediction for new input x:

Let predictions = []

For each tree Ti in RF:

Predict class label ci = Ti.predict(x)

Append ci to predictions

Return the class with majority vote in predictions

5.2.2 XGboost

The word XGBoost means Extreme Gradient Boosting, which is an advanced and fast gradient-boosting algorithm with applied debugging procedures to also improve practical utility. This algorithm sequentially builds decision trees, each correcting the errors of the previous trees based on predictors. It will take a weak learner and build it with others into a strong learner, incorporate regularization methods, and optimize it for better generalization under boosting.

Most importantly, the typical attribute of XGBoost is that these first-order gradients can even be modeled with a further higher probability. However, it must be emphasized that just like the second-order gradients which are referred to as Hessians, they would result in very phenomenal accuracy and even improved robustness. The learning rate or shrinkage operation is another characteristic of the XGBoost algorithm where the column subsampling reduces overfitting and improves the overall diversity of the trees. Thus, it would save a lot of time wasted while training all those jobs, thus enabling the real-time application at a large scale.

Thus, here is the classification of URLs inside XGBoost. URLs are classified into prediction outputs that can be utilized in the phishing detection project according to features that are coded from the suspicious patterns to letters and symbols together with other domain characteristics. In simple words, "it analyzes the different input features and measures against a classification (URL or phishing) as the condition of the outcome variable. This is because of the ensemble classification technique involved in the project since it has very high predicting power and importance ranked feature.

The attributes of XGBoost as classifiers become even more beneficial to solve such problems as detection of phishing attacks. The tuning parameters give XGBoost the very body of ensemble leaning while handling the easiest at the other end in tuning speed. And, thus, it becomes usable for real-time threat-detection systems because it can be tuned when the strategies of the perpetrator vary on offense.

Algorithm:

Input:

Training data (URLs with labels)

Number of boosting rounds (N)

Learning rate (η)

Initialize:

Model prediction $F_0(x) = 0$

For i = 1 to N:

1. *Compute pseudo residuals:*

   r_i = y_i - F_{i-1}(x_i) // Error of current prediction

2. *Fit a regression tree h_i(x) to residuals r_i*
3. *Compute weight γ for each leaf of the tree to minimize loss*
4. *Update the model:*

   *F_i(x) = F_{i-1}(x) + η * h_i(x)*

End For

Final Prediction:

Sign(F_N(x)) or threshold-based output (e.g., phishing if score > 0.5)

5.2.3 ENSEMBLE MODEL

The ensemble model employs different machine learning classifiers, applied to be accessed for improved industrial output predictions in phishing detection, is used to increase the efficiency of detection and accuracy of the phishing system. The contribution of multiple connected model predictions to the production of safe and generalizable output in ensemble learning systems should replace the output produced through single model deployment. So, successfully preventing overfitting has led to the development of more accurate models that manage complex or unbalanced datasets on phishing data.

In this project, the voting ensemble technique is applied to Random Forest and XGBoost, two of the most potent classifiers. This is because Random Forest takes care of datasets having multiple dimensions and prevents overfitting through the results of several decision trees being averaged, while XGBoost makes effective and accurate performances in gradients boosted mechanisms against structured types of data. Such merging results in a model that performs well and remains stable during its process.

In this framework, combined system utilizes soft voting as its ensemble strategy where models are pooled in predicting whether a URL is phishing or legitimate. The actual output is the averaged probabilities such that the highest one ultimately decides its classification threshold will thus conclude more informed and accurate final decisions than by the mere majority class vote protocols through such analyses of individual classifier confidence levels.

The combined application brings out the benefits of Random Forest generalization along with the optimization of XGBoost in the scope of phishing URL detection. The combined model delivers better results in detecting phishing as fraud detection owing to lower false positive rates and better accuracies, which really matters in the real-time security systems and particularly in browser extensions against phishing protection.

Algorithm:

Input:

Training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}

Base classifiers: Random Forest (RF), XGBoost (XGB)

Step 1: Train each base model independently

Train RF on D

Train XGB on D

Step 2: Make predictions on test data

For each input sample x in test set:

prob_RF = Predict probabilities using RF

prob_XGB = Predict probabilities using XGB

Step 3: Combine predictions

Final_Prob = (prob_RF + prob_XGB) / 2

Final_Prediction = Class with highest Final_Prob

5.3 CHROME EXTENSION DEVELOPMENT

Users can get alerts about real-time phishing URLs through the Chrome extension which functions as the interface while browsing. The tool works through browser URL monitoring that sends this information to analyze on the backend server. The complete process operates in the background without interrupting browsing activity so users receive instant feedback while browsing online. The frontend part of the extension works with HTML and CSS and JavaScript to provide users an interactive interface that can work on contemporary browser environments.

The Chrome extension retrieves accessed URLs that get transmitted to the Flask backend server through HTTP requests. The backend analyzes features of the URL by making use of its hosted phishing detection model to provide classification results. The analysis result by the backend feeds into a straightforward alert notice to users disclosing phishing or legitimate status of the URL. The system allows users to access immediate output that helps them determine the potential dangers of specific links.

The extension design maintains security essentials as well as usability features. Running persistently in the background function enables secure protection alongside maintenance of browser performance which provides users with encryption services while they stay unaware of any lag. Multiple visual elements and alerts generation from the extension provide users with information regarding URL risk which enables anyone regardless of computer skills to understand the results. The extension provides good security protection against malicious phishing websites.

The Chrome extension features built-in modularity which simplifies process updates for backend model upgrades and lets users incorporate new features including blocking specific URLs or site reputation assessments or user interaction feedback. The extension maintains its effectiveness as an anti-phishing tool through its ability to scale up when facing new and emerging security threats.

Users interact with the phishing detection system through the Chrome extension which provides them access to backend machine learning model

intelligence. The chrome extension allows users to securely browse through the websites by providing real-time phishing alerts when inputting a URL using the phishing detection system.

5.4 INTEGRATION OF PHISHING MODEL WITH CHROME EXTENSION

The phishing detection model combined with Chrome extension leads to crucial real-time analysis capabilities that protect users during their internet browsing activities. The ensemble model (Random Forest and XGBoost) is placed in the backend server for processing while connecting through the browser extension. The Chrome extension functions as the interface between users and performs real-time URL collection that it sends to the backend analysis engine.

The Chrome extension gathers the currently visible webpage URL through JavaScript APIs when a user types the URL or pastes it. A Flask API provides the mechanism to send the URL from the frontend to the backend Python server by making an HTTP POST request. The backend handles this request by extracting URL features which include IP addresses together with special characters and domain age information along with other phishing indicators. The trained ensemble model receives these features to establish whether the URL contains phishing content.

The processed classification output is transmitted to the extension by the model. The system shows this result instantly to users by presenting warning alerts about dangerous links. The system operates at high speed to prevent delays that might occur during web browsing. The protection system functions as a background operation while remaining lightweight which allows uninterrupted workflow for users. The deployment process for the chrome extension alongside the integration of the flask server appears in Fig A2.7 to Fig A2.10. Real-time URL predictions become classified as the chrome extension records the URL before sending it to the flask server.

The integrated system safeguards users by alerting the users and preventing them from accessing malicious sites in advance. The system delivers machine learning capabilities straight to the end user through their browser in order to protect their security. The system unites a powerful detection model for

phishing attacks with browser-based interaction to create an automated security protection system which works in real time.

5.4.1 Flask Server

The Flask server functions as the main backend processing tool which analyzes phishing detection requests within the project framework. Flask represents an excellent web framework option because it is written in Python to deliver lightweight functionality for building strong yet simple web applications and APIs. Real-time phishing URL analysis occurs through an HTTP-based interface that links the machine learning model to the Chrome extension due to the usage of Flask in this project. Its straightforward design with adjustable components together with efficient ML workflow compatibility positions Flask as an appropriate platform for developing web services from ML models.

A POST request originates from the Chrome extension to the Flask server after the extension collects browser URLs from users. The server receives incoming requests that send processed data which usually contains the URL that needs analysis. The Flask route (@app.route) receives requests which begin by extracting required input data before triggering both preprocessing and feature extraction functions. The system design enables uninterrupted data processing starting from initial raw URL entry concluding in an ensemble model prediction result.

The Flask application employs joblib and pickle for loading the pre-trained machine learning models Random Forest and XGBoost. After extracting the URL features from the input they will be processed by the ensemble model for prediction. The system produces a prediction indicating whether the input URL belongs to the “phishing” category or the “legitimate” category. The response returns as JSON data to the Chrome extension from the application. Through its Flask server the system maintains speedy operations for delivering on-the-spot feedback to users who need immediate alerts about risky websites.

The Flask server serves as a fundamental element that connects the Chrome extension visible to users with the machine learning backend

functionality. The Flask server functions as a communication link that connects requests from the front end to back-end processes to deliver quick and protected phishing detection capabilities. The server can be utilized either locally or on the cloud through simple deployment methods which enables flexible hardware scalability for upcoming features including user interaction logging as well as additional security checks integration.

CHAPTER 6

RESULTS AND ANALYSIS

6.1 EVALUATION METRICS

The evaluation of the phishing URL classification system requires proper assessment techniques which enable precise measurement of ensemble model performance. This project uses four major evaluation metrics including Accuracy, Recall, Precision as well as F1 Score. The assessment metrics used to measure the model behavior enable model behavior assessment while offering direction for strengthening the system for phishing threat detection.

Accuracy: The model's accuracy is calculates as shown in Eq.1 evaluates its performance through a measurement of correct predictions against total predictions. The metric offers an overview of the model performance quality for all categories.

$Accuracy=(TP+TN)/P+N)$ ----------->(Eq. 1)

Where
TP = True Positives,
TN = True Negatives,
P = the total number of positives, and
N = the total number of negatives.

The metric presents a general success percentage for the model to detect both types of URLs (legitimate and phishing).

Recall (also known as Sensitivity): The detection capability of actual phishing URLs constitutes Recall because it deals with Sensitivity. It is measured by Eq.2 Successful detection of phishing links remains crucial in this project since failure to identify a phishing link may result in severe security problems.

$Recall = TP/(TP+FN)$ ----------->(Eq. 2)

The detection performance improves when most actual phishing URLs receive identification warnings from the model.

Precision demonstrates the accuracy of positive predictions made by the model as shown in Eq.3. Within a phishing detection system the measurement represents the number of correctly identified phishing links among all recognized phishing URLs.

$Precision = TP/(TP + FP)$ ----------> (Eq. 3)

This performance measure stands as vital when minimizing false alert situations because it helps prevent blocking of authentic web pages unnecessarily.

F1 Score functions as a balanced metric because it combines Precision and Recall to create its evaluation. The calculation of F1-score is shown in Eq.4 This evaluation metric works best for situations that combine the need to avoid phishing link mistakes with the avoidance of incorrect safe URL identification.

$F1\ Score = 2 \times [(Precision \times Recall)/(Precision + Recall)]$ ----------->(Eq. 4)

A single numerical value helps users determine how reliably their model classifies phishing URLs.

Both accuracy and reliability in practical scenarios become evident through multiple performance indicators which assess model performance in phishing detection tasks. These metrics create a basis for enhancing models while preparing them to become operational.

6.2 PHISHING DETECTION

The evaluation process for the phishing detection model used different testing techniques to capture its reliability and robustness conditions. Multiple real-time evaluations of the model occurred with datasets which included phishing as well as legitimate URLs. The classification performance of predictions received assessment through standard metrics such as Precision and Accuracy along with Recall and F1 Score to determine correct input identification.

An evaluation of the model's simulated deployment performance was conducted using PowerShell's API testing protocols. The Flask REST API served the model for receiving input URLs through PowerShell scripts which executed automated HTTP requests. The system stored record tracking data

which included response comparisons to defined labels for predicting outcome results. Testing of several URLs at once became possible through this system which enabled assessment of the model's real-time performance.

Testing utilized this approach simultaneously to evaluate predictive strength and backend API reliability as well as API response speed. Implementing PowerShell for automated testing expanded the system evaluation to situations that resembled practical use cases including browser extensions. Real-time operational testing of the phishing detection system validated its capability to deploy effectively by providing precise and dependable predictions for electronic safety solutions.

6.2.1 Model Observations

The phishing detection system consistently performed quite good in all of the major evaluation testing metrics. Model recall when high means that it can find all phishing URLs successfully from showing examples. All F1 scores are equally balanced, proving that the model is capable of equally delivering excellent precision and recall for different input examples. The precision-confidence curve gives indication about the ability of the model to create inferences from incorrect detections of safe URLs thus protecting user experience in real-world scenarios.

- **Feature Extraction Comparison**

  In this study three feature extraction techniques were evaluated, i.e. TF-IDF (Term Frequency-Inverse Document Frequency), PCA (Principal Component Analysis) and Artificial Neural network (ANN)-based feature encoding.TF-IDF was selected owing to efficiency in extraction of text features from the URLs capturing both term frequencies and uniqueness. PCA was utilized for dimensionality reduction, particularly on numerical or embedded URL features. ANN encoding was the other source wherein an automated learning of feature representation was sought. Among these as shown in Fig 6.1, TF IDF stood out as the most performing detection accurate and computationally efficient method for single detection task making it by far preferable for the task at hand.

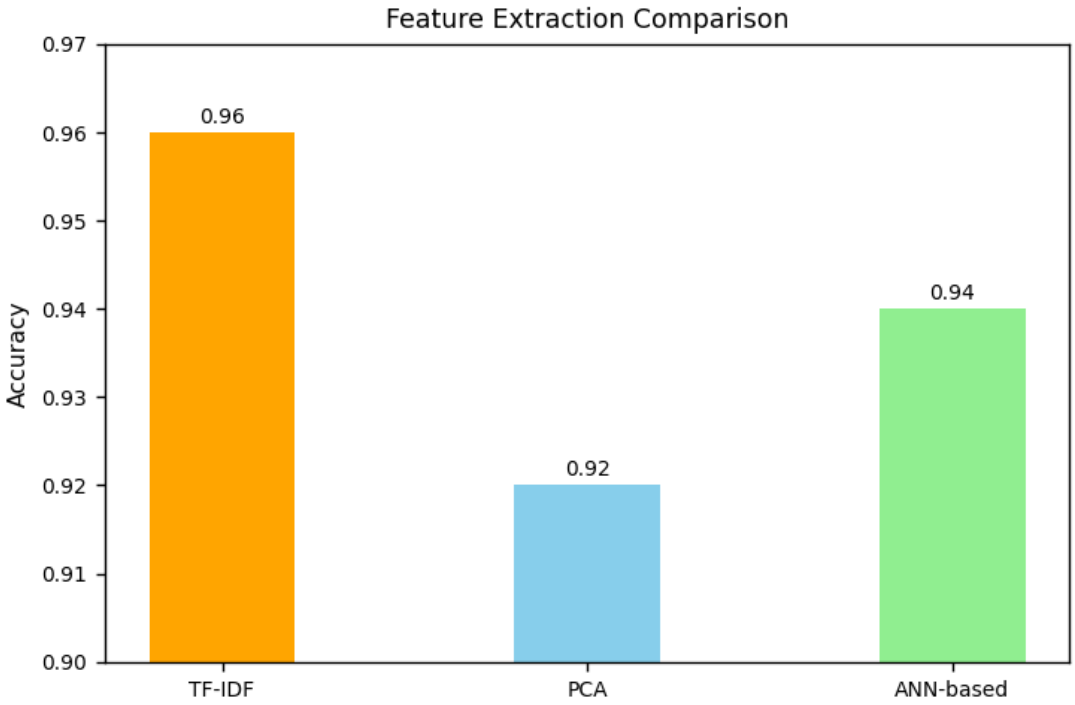

Fig 6.1 Feature Extraction Model Comparison

- **ENSEMBLE MODEL COMPARISON**

  Fig. 6.2 presents a combination of models evaluated for classification: Logistic Regression, Decision Trees, Random Forest, and XGBoost. While stand-alone models appear to perform reasonably well, it is the combination of Random Forest and XGBoost in an ensemble approach that has yielded the best results.

  The Multi-Layer Stacked Ensemble Learning Model (MLSELM) takes advantage of the strengths of both these base classifiers, allowing for significant overall accuracy improvement while minimizing false positives. The model also adapts more easily and accurately to the varying patterns of phishing and captures the complex relationships and patterns present in the data.

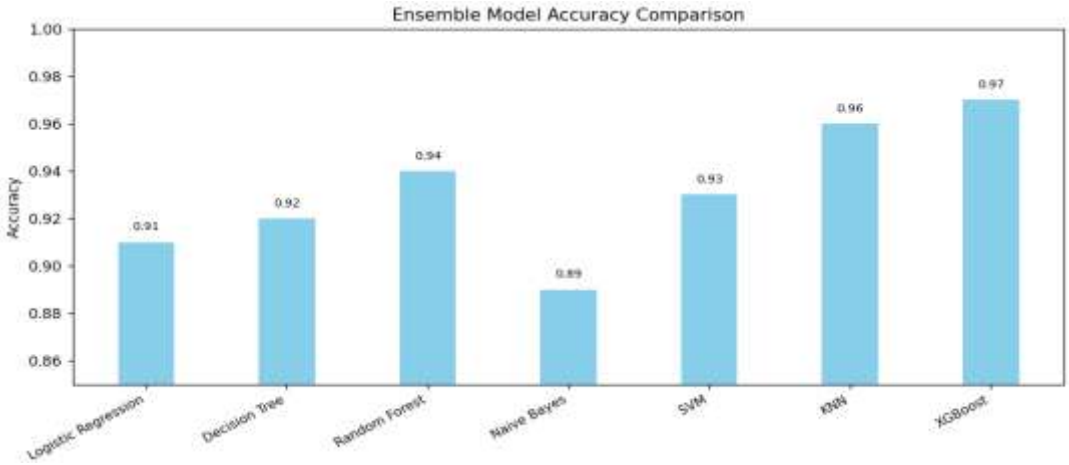

Fig 6.2 ML Model Comparison

- **Confusion Matrix**

  The provided matrix gives detailed information about the model's ability to distinguish between phishing and legitimate links. A model demonstrates successful performance through its distribution of true positives and true negatives observed along the diagonal area of the matrix. The modeling mistakes can be located in the elements that lie outside the main diagonal of the confusion matrix. These classification errors provide essential information to evaluate the system's capability for both false attack recognition and unidentified attacks that would occur in deployed scenarios.

  The proposed phishing detection system demonstrates strong correlations between real-world and predicted results through the information displayed in Fig 6.3. The ensemble model which combines XGBoost with Random Forest shows strong reliable performance based on the diagonal value concentration. The system demonstrates its operational efficacy through these results because it achieves effective malware URL detection with minimal false positive output which boosts online security for users during surfing activities.

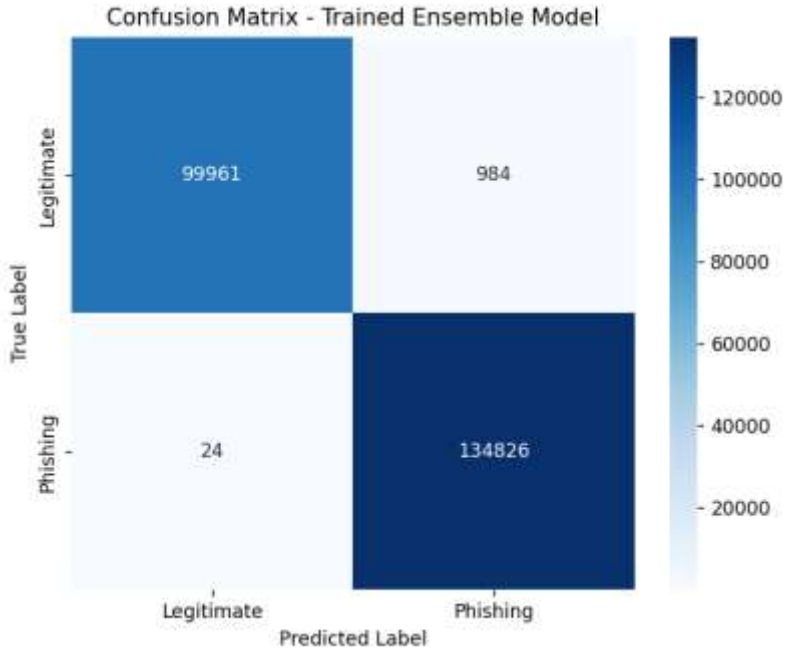

Fig 6.3 Confusion Matrix

- **Accuracy**

  The analysis of accuracy on a class basis reflects the effectiveness of a phishing detection model in correctly classifying the URLs between legitimate and phishing URLs. The ensemble model, which uses XGBoost and Random Forest together, is high in the classification accuracies consistently across both the classes. It shows excellent potential for correct predictions concerning the true nature of any incoming URLs, thus promising minimal misclassification in real-world applications. The system shows itself reliable for real-time browser-based phishing detection using strong accuracy in identifying safe and malicious web links. This feature meets practical deployment scenarios like a Chrome extension.

- **Precision**

  As shown in Fig 6.4, the precision performances of the phishing detection model kept the same level for phishing and real classes of URLs. The model has kept high-quality true positives at all its positive predictions. High precision proves vital for phishing detection systems as it prevents legit URLs from being confused with false ones.

The same consistency in precision allows fast and accurate decisions necessary to keep user disturbances minimal that this system can retain its utility in real-world applications such as Chrome extensions.

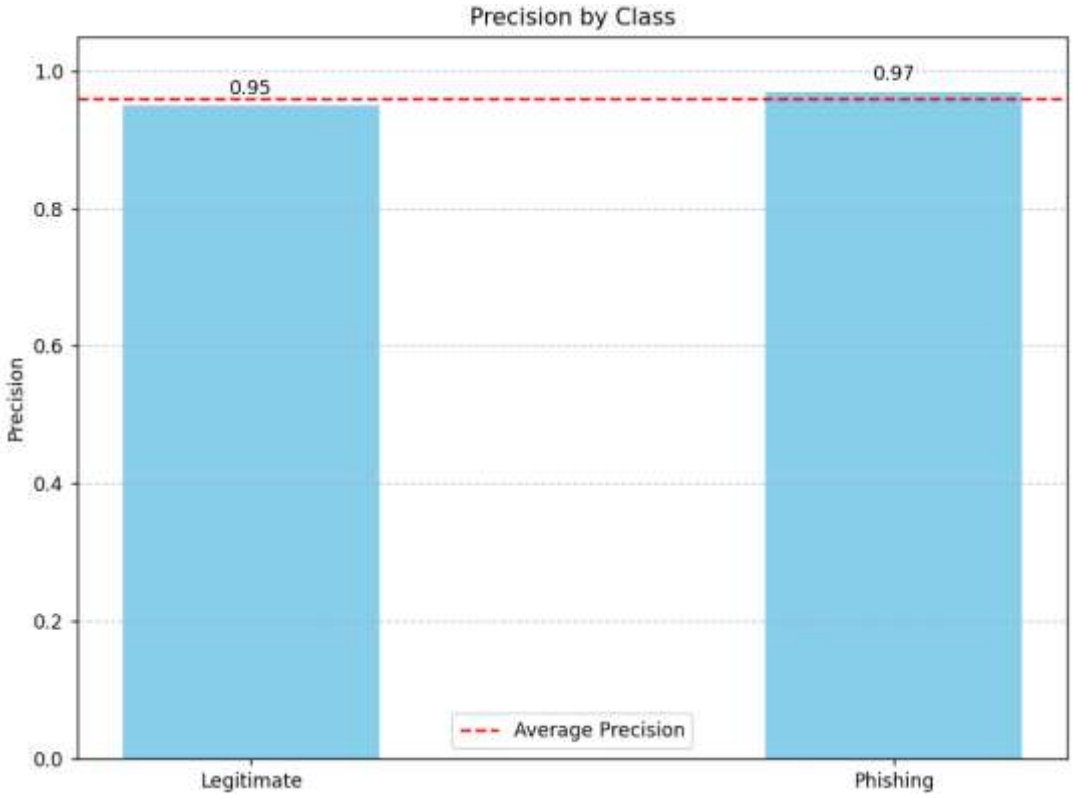

Fig 6.4 Precision

- **Recall**

  Recall analysis as shown in Fig 6.5, the model has proved almost perfect for identifying both real phishing and real legitimate URLs. High recalls hold special value for phishing detection systems since these reduce false negatives and thus missed threats in general. The model performs very well in many situations of input, which makes it fit for the real operation environment. This brings improvement to web safety, as such high recall capability within the Chrome extension warns the user directly against malicious sites.

Fig 6.5 Recall

- **F-1 Score:**

  As observed in Fig 6.6 through activities, F1 score analysis provides the optimum balance between precision and recall metrics for each activity group. It shows that the model maintains a proper accuracy wherein positive cases can be detected with minimal permit false positives which indeed reflects upon its strong capacity to positively classify multiple human activities.

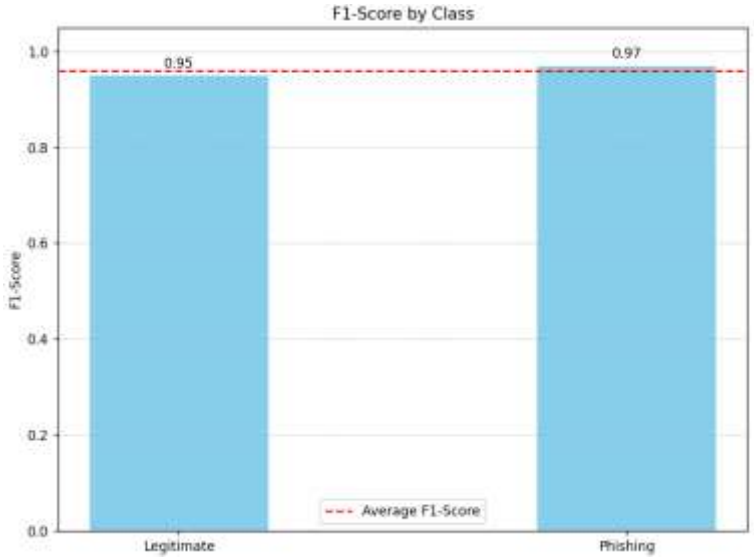

Fig 6.6 F1-Score

6.3 CHROME EXTENSION

The extension operates automatically in users' browser setting. The extension obtains active tab URLs to pass them to a Flask-based API that contains trained XGBoost and Random Forest ensemble model. The server operates the URL through an HTTP POST request then activates the TF-IDF vectorizer for identifying the webpage as either "phishing" or "legitimate." The popup interface displays the real-time result to notify users about the webpage classification.

A comprehensive test regimen was applied to the extension because it required meeting high performance criteria in terms of usability along with quick response and precise outcomes. URL entry workflow lasted less than one second before the system displayed predictions through a polished user interface. Various tests across different browsing sessions and websites demonstrated that the extension successfully supported various URL requests while maintaining uninterrupted server connectivity which ensured smooth browsing activities.

The Chrome Extension contributes to higher practical usability as well as expanded accessibility of the phishing threat detection system. Venue-based users gain proactive security feedback through built-in browser AI classification technology which operates without manual process or data input. The implementation of the machine learning model as an active and real-time defense mechanism through integration transforms it from being a backend tool against phishing attacks, making it ideal for both individual and enterprise use.

6.4 INTEGRATION OF PHISHING MODEL AND CHROME EXTENSION:

The phishing detection model linked to Chrome extension interface merges both machine-learning capabilities with instant user interaction to increase web-safety protection. An ensemble model which combines XGBoost and Random Forest classifiers forms the core of a system developed to accurately identify between authentic URLs and phishing URLs.

A deployment of this trained model takes place through Flask API integration which operates with a seamless connection to the Chrome extension frontend by using JavaScript and HTTP request management.

Customer input through the extension leads the backend system to conduct a text processing operation using the TF-IDF vectorizer before sending the URL for classification into "Phishing" or "Legitimate" categories by the ensemble model. After users enter the URL in the input field the system shows instant feedback which indicates whether the website is safe or not below the input field. The instantaneous processing system operates like real-world deployments therefore it provides users the opportunity to make safe decisions before accessing suspicious sites.

Users gain access to an effective simple security solution against phishing attacks when machine learning works together with browser-based interaction. The present work establishes a framework which enables future additions such as suspicious URL recording and feedback collection as well as speech-based warning integration to extend system applications into more extensive cybersecurity functions.

6.5 RESULTS AND DISCUSSION

The below section discuss the outcomes and results of the phishing detection project such as evalutation metrics and chrome extension output.

Evaluation Metrics

A detection model for phishing using PowerShell API calls along with URL classification has been designed and evaluated by a research group. The excitement mainly lies in the ensemble modeling aspect, which subsequently allows the live system ingesting input data in real time for predictions. Performance was evaluated against the various phishing threats in terms of common evaluation metrics, thereby arriving at accuracy, recall, and precision, which were then combined for an F1 score to provide a more rounded view of effectiveness.

By and large, this is a massive step forward. Very high accuracy proves their reliability in strong and good performance predicting PowerShell activity.

While modeling and verification, we kept precision and recall-the ability of the model, in real-world situations, not to raise false alarms-false positives-while grabbing actual threats-false negatives-as the most important. The F1 score once again confirmed ensemble models involving the XGBoost and Random Forest classifiers in a balanced way, providing meaning to a more integrated view of their performance.

The visual representation of results in Fig 6.7 and Fig 6.8 shows that the phishing detection system was able to maintain stable performance on different sets of test samples.

The diagram shows URL inputs from different batches performing on metrics that show that the system is really capable of maintaining stable performance under the demands of real-time web-based phishing detection tasks. This output indicates the model is ready for deployment on web-based systems like browser extensions for Chrome.

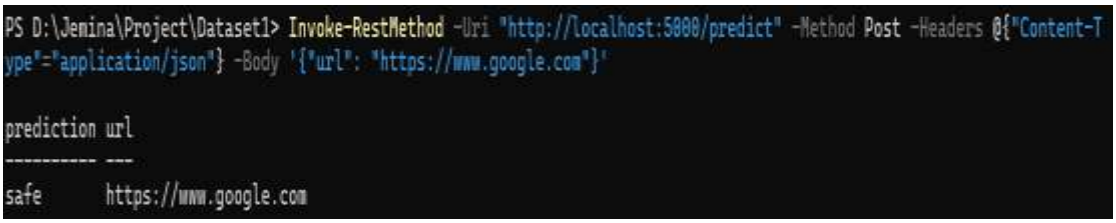

Fig 6.7 Powershell API testing for safe website

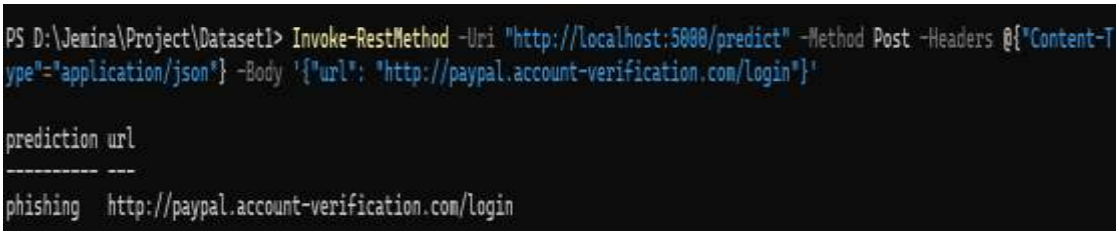

Fig 6.8 Powershell API testing for phishing website

Chrome Extension

While using the extension users receive instantaneous phishing detection results during URL input to the text box. When a user begins entering URLs into the extension the data sends to a Flask-based backend server that uses a TF-IDF vectorizer to preprocess the input before the ensemble model with XGBoost and Random Forest classifiers processes it.

The extension reveals the prediction result of the entered URL as "Phishing" or "Legitimate" immediately after detection directly beneath the input box. An immediate feedback system through this mechanism warns users about dangerous websites prior to their website navigation. The system shows the output results through clear and visible color indicators which display red for identified phishing URLs and green for non-phishing URLs. The extension uses asynchronous JavaScript to automatically generate the response results without any need from the user to submit the form.

The ensemble model correctly labels websites either "Phishing" for dangerous URLs or "Legitimate" for secure URLs as depicted in Figure 6.9 and Fig 6.10. The result presentation system indicates that the system shows potential to serve as an effective practical program which protects users from phishing websites through browser platforms

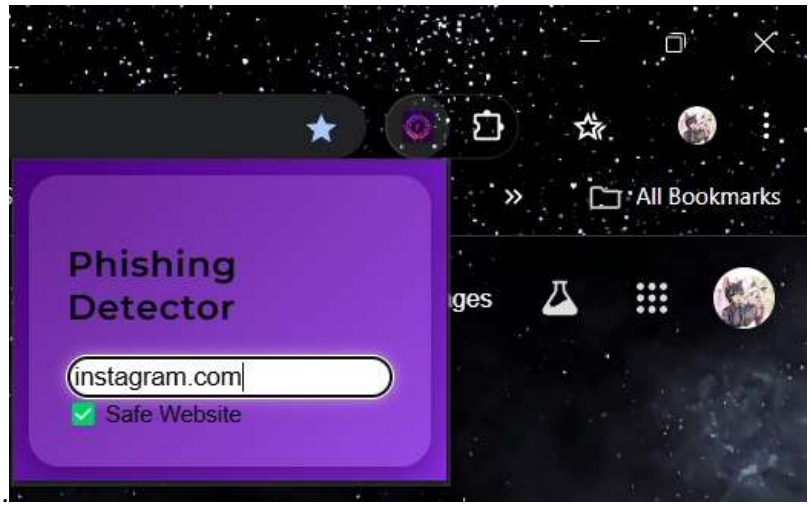

Fig 6.9 Chrome Extension for detecting safe website

Fig 6. 10 Chrome Extension for detecting phishing website

CHAPTER 7
CONCLUSION AND FUTURE WORK

Phishing attacks have become the leading dangerous digital threat which affect today's cyber-world. The existing blacklist filters alongside signature-based detection and static rule engines fail to detect newly created phishing websites that keep avoiding the detection protocols. Preventive methods based on established patterns or databases create gaps when protecting users from newly generated phishing threats during online sessions.

This project develops an innovative solution to overcome existing traditional methods through the creation of a fast and real-time phishing detector that works as a Google Chrome extension. The extension enables instant website legitimacy checks because it includes machine learning models directly within Chrome to verify any URL users provide. A dual-classifier system made of XGBoost and Random Forest succeeded in automatic URL classification with a steady accuracy of 95-98% from a balanced data sample including both phishing and true URLs.

The user-centric design with deployment capabilities makes this system stand out as its central feature. The embedded model inside the Chrome extension analyzes the user input URL instantly after the extension detects the URL entry into its input box. After entering a URL into the field a red or green warning appears instantly next to it to show phishing status. The real-time response system improves the user's view and choice-making skills because it activates through the browser interface's security layer. Live environments reveal the system features few incorrect positive and negative detections because of its dependable and reactive performance according to output figures.

This project provides users with an efficient browser-based solution which enables immediate detection of phishing attacks while they use the system. The system prevents users from needing outside assistance or delayed actions by delivering instant and intelligent threat detection abilities directly to them. The proposed system demonstrates potential for enterprise-scale security together with school and organizational secure browsing protection and future implementation of real-time link screening and automatic alert functionality.

REFERENCES

[1] L. R. Kalabarige, R. S. Rao, A. Abraham, and L. A. Gabralla, "Multilayer stacked ensemble learning model to detect phishing website," IEEE Access, vol. 80, pp. 3178–3183, 2022.

[2] G. S. Nayak, B. Muniyal, and M. C. Belavagi, "Enhancing phishing detection: A machine learning approach with feature selection and deep learning models," IEEE Access, vol. 13, Feb. 2025.

[3] W. Guo, Q. Wang, H. Yue, H. Sun, and R. Q. Hu, "Efficient phishing URL detection using graph-based machine learning and loopy belief propagation," IEEE, Jan. 2025.

[4] M. A. Daniel, S.-C. Chong, L.-Y. Chong, and K.-K. Wee, "Optimising phishing detection: A comparative analysis of machine learning methods with feature selection," J. Informatics Web Eng., Feb. 2025.

[5] O. K. Sahingoz, E. Buber, and E. Kugu, "DEPHIDES: Deep learning based phishing detection system," IEEE Access, vol. 12, Jan. 2024.

[6] R. R. Raj, R. P. Remya, and A. Abraham, "Phishing URL detection using residual multi-layer perceptron (ResMLP) model," IEEE Access, vol. 15, pp. 1221–1235, 2024.

[7] A. S. Rafsanjani, M. H. Anisi, A. A. A. Ghani, and N. Ahmad, " Enhancing Malicious URL Detection: A Novel Framework Leveraging Priority Coefficient and Feature Evaluation," IEEE Access, vol. 11, pp. 115982–115996, 2024.

[8] Wenhao Li, S. Manickam, Y.-W. Chong, W. Leng, and P. Nanda, “A state-of-the-art review on phishing website detection techniques,” IEEE Access, vol. 12, Dec. 2024.

[9] D. J. Dsouza, A. P. Rodrigues, and R. Fernandes, "Multi-modal comparative analysis on execution of phishing detection using artificial intelligence," IEEE Access, vol. 12, Nov. 2024.

[10] M. Almousa and M. Anwar, "A URL-based social semantic attacks detection with character-aware language model," IEEE Access, vol. 11, Jan. 2023.

[11] A. Karim, S. B. Belhaouari, M. Shahroz, K. Mustofa, and S. R. Joga, "Phishing detection system through hybrid machine learning based on URL," IEEE Access, vol. 11, Mar. 2023.

[12] A. R. Pais, P. U. Krishnan, and M. M. Pai, "Detection of phishing websites using boosting-based hybrid feature selection and multilayer stacked ensemble learning model," IEEE Access, vol. 225, pp. 312–319, 2023.

[13] S. Belhaouari, A. B. Hamou, A. N. AlZain, and S. Khan, "A hybrid machine learning model for phishing detection based on URL features," IEEE Access, vol. 74, no. 1, pp. 1751–1768, 2023.

[14] A. Ozaday, E. Duman, and M. T. Ozsu, "Phishing URL detection using machine learning algorithms: Random Forest achieves highest accuracy," IEEE Access, vol. 72, pp. 103419, 2023.

[15] R. Zieni, L. Massari, and M. C. Calzarossa, "Phishing or not phishing? A survey on the detection of phishing websites," IEEE Access, vol. 11, ,2023.

[16] M. Sánchez-Paniagua, A. Guzmán-Serrano, and R. García-Hernández, "Phishing detection system based on login URL patterns using machine learning models," IEEE Access, vol. 11, pp. 104345–104358, 2022.

[17] B. T. Mummadi and N. Puligundla, "Detection of phishing websites using supervised learning," Int. J. Intell. Syst. Appl. Eng., 2022.

[18] M. Ok, I. Kara, and A. Ozaday, "Characteristics of understanding URLs and domain names features: The detection of phishing websites with machine learning methods," IEEE Access, vol. 10, Nov. 2022.

[19] M. Sameen, M. M. Rathore, and A. Paul, "PhishHaven: A deep learning-based real-time detection system for phishing websites," IEEE Internet of Things Journal, vol. 10, no. 2, pp. 1341–1350, Jan. 2020.

[20] S. Bell and P. Komisarczuk, "An analysis of phishing blacklists: Google Safe Browsing, OpenPhish, and PhishTank," in Proc. Australas. Comput. Sci. Week Multiconf. (ACSW), Melbourne, VIC, Australia. New York, NY, USA: Association for Computing Machinery, 2020, Art. no. 3

APPENDIX

A1 - SOURCE CODE

PHISHING MODEL

The model begins its operation by deriving features from TF-IDF vectorized URL text to generate useful numerical data. The ensemble model comprised of XGBoost and Random Forest classifiers receives training with this information which produces exceptional results during detection of phishing and legitimate URLs.

DATA PREPROCESSING

```
import pandas as pd
import scipy.sparse

dataset1_path = 'PhiUSIIL_Phishing_URL_Dataset1.csv'

def clean_and_extract_features(dataset_path,
cleaned_dataset_path, feature_extraction_path):
    df = pd.read_csv(dataset_path, engine='python')
    df = df.drop_duplicates()
    df = df.dropna()
    target_column = 'phishing' if 'phishing' in df.columns else
'label' if 'label' in df.columns else 'status'
    if target_column not in df.columns:
        raise ValueError("No valid target column found in
dataset")
    y = df[target_column].map({'legitimate': 0, 'phishing': 1}) if
target_column == 'status' else df[target_column]
    url_column = 'URL' if 'URL' in df.columns else 'url' if 'url' in
df.columns else None
    if not url_column:
        raise ValueError("No URL column found in dataset")
```

```
    df.to_csv(cleaned_dataset_path, index=False)
    print(f"Cleaned dataset saved to {cleaned_dataset_path}")

    from sklearn.feature_extraction.text import TfidfVectorizer
    df_cleaned = pd.read_csv(cleaned_dataset_path)
    url_texts = df_cleaned[url_column].astype(str)
    tfidf_vectorizer = TfidfVectorizer(max_features=200)
    X_tfidf = tfidf_vectorizer.fit_transform(url_texts)
    scipy.sparse.save_npz(feature_extraction_path, X_tfidf)
    print(f"Feature extracted dataset saved to
{feature_extraction_path}")

clean_and_extract_features(dataset1_path,
'cleaned_dataset1.csv', 'feature_extracted1.npz')
clean_and_extract_features(dataset2_path,
'cleaned_dataset2.csv', 'feature_extracted2.npz')
```

FEATURE EXTRACTION

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import joblib
dataset_path = "cleaned_dataset.csv"
df = pd.read_csv(dataset_path)

url_column = 'url' if 'url' in df.columns else 'URL'
url_texts = df[url_column].astype(str)
tfidf_vectorizer = TfidfVectorizer(max_features=500)
X_tfidf = tfidf_vectorizer.fit_transform(url_texts)
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')
print("Feature extraction complete. TF-IDF vectorizer saved as
tfidf_vectorizer.pkl")
```

PHSIHING MODEL:

```
import pandas as pd
import numpy as np
import joblib
from sklearn.ensemble import RandomForestClassifier,
VotingClassifier  # Fixed import
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
classification_report

dataset_path = "cleaned_dataset.csv"
df = pd.read_csv(dataset_path)

url_column = 'url' if 'url' in df.columns else 'URL'
target_column = 'phishing' if 'phishing' in df.columns else
'label'

tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')
X_tfidf = tfidf_vectorizer.transform(df[url_column].astype(str))

y = df[target_column].map({'legitimate': 0, 'phishing': 1}) if
target_column == 'status' else df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y,
test_size=0.2, random_state=42)

xgb = XGBClassifier(eval_metric='logloss')
rf = RandomForestClassifier(n_estimators=100,
random_state=42)

ensemble_model = VotingClassifier(estimators=[('xgb', xgb),
('rf', rf)], voting='soft')
ensemble_model.fit(X_train, y_train)

y_pred = ensemble_model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)

print(f"Ensemble Model Accuracy: {accuracy:.4f}")
print("Classification Report:\n", classification_report(y_test,
y_pred))

joblib.dump(ensemble_model,
'phishing_ensemble_model.pkl')
print("Model saved as phishing_ensemble_model.pkl")
```

EVALUATION METRICS

Below are the evaluation metrics for the trained phishing dataset which consists of Confusion Matrix, Accuracy, Precision, Recall and F1-Score

CONFUSION MATRIX

```
import pandas as pd
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

df = pd.read_csv("cleaned_dataset.csv")
print("Columns:", df.columns)
url_column = 'url' if 'url' in df.columns else 'URL'
target_column = 'phishing' if 'phishing' in df.columns else
'label'

tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')
X_tfidf = tfidf_vectorizer.transform(df[url_column].astype(str))

y = df[target_column]
model = joblib.load('phishing_ensemble_model.pkl')
y_pred = model.predict(X_tfidf)

cm = confusion_matrix(y, y_pred)
labels = ['Legitimate', 'Phishing']
```

```
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix - Trained Ensemble Model")
plt.tight_layout()
plt.show()
```

PRECISION

```
import pandas as pd
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv("cleaned_dataset.csv")
model = joblib.load("phishing_ensemble_model.pkl")
vectorizer = joblib.load("tfidf_vectorizer.pkl")

url_column = 'url' if 'url' in df.columns else 'URL'
target_column = 'phishing' if 'phishing' in df.columns else
'label'
X = vectorizer.transform(df[url_column].astype(str))
y = df[target_column].map({'legitimate': 0, 'phishing': 1}) if
target_column == 'status' else df[target_column]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
y_pred = model.predict(X_test)

report = classification_report(y_test, y_pred,
target_names=["Legitimate", "Phishing"], output_dict=True)
classes = ["Legitimate", "Phishing"]
```

```
precision = [report[cls]['precision'] for cls in classes]

plt.figure(figsize=(8, 6))
bars = plt.bar(classes, precision, color='skyblue', alpha=0.7)
plt.title("Precision by Class")
plt.ylabel("Precision")
plt.ylim(0, 1.05)
plt.axhline(y=np.mean(precision), color='red', linestyle='--',
label='Average Precision')
plt.legend()
for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2,
bar.get_height()+0.01, f"{bar.get_height():.2f}", ha='center')
plt.tight_layout()
plt.show()
```

RECALL

```
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

df = pd.read_csv("cleaned_dataset.csv")
tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')
model = joblib.load('phishing_ensemble_model.pkl')

url_column = 'url' if 'url' in df.columns else df.columns[0]  #
fallback to first column if needed
target_column = 'phishing' if 'phishing' in df.columns else
'label'

X = tfidf_vectorizer.transform(df[url_column].astype(str))
y = df[target_column].map({'legitimate': 0, 'phishing': 1}) if
target_column == 'status' else df[target_column]
```

```
y_pred = model.predict(X)

report = classification_report(y, y_pred, output_dict=True,
target_names=['Legitimate', 'Phishing'])

classes = list(report.keys())[:2]
recall = [report[cls]['recall'] for cls in classes]

plt.figure(figsize=(8, 6))
bars = plt.bar(classes, recall, color='lightgreen', alpha=0.7)
plt.title("Recall by Class")
plt.ylabel("Recall")
plt.ylim(0, 1.05)
plt.axhline(y=np.mean(recall), color='red', linestyle='--',
label='Average Recall')
plt.legend()

for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2,
bar.get_height()+0.01, f"{bar.get_height():.2f}", ha='center')
plt.tight_layout()
plt.show()
```

F1-SCORE

```
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

df = pd.read_csv("cleaned_dataset.csv")
url_column = 'url' if 'url' in df.columns else 'URL'
target_column = 'phishing' if 'phishing' in df.columns else
'label'
```

```
tfidf_vectorizer = joblib.load('tfidf_vectorizer.pkl')
ensemble_model =
joblib.load('phishing_ensemble_model.pkl')  # or your model
name

X = tfidf_vectorizer.transform(df[url_column].astype(str))
y = df[target_column].map({'legitimate': 0, 'phishing': 1}) if
target_column == 'status' else df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y,
stratify=y, test_size=0.2, random_state=42)
y_pred = ensemble_model.predict(X_test)

report = classification_report(y_test, y_pred,
target_names=['Legitimate', 'Phishing'], output_dict=True)
classes = list(report.keys())[:2]  # Only 'Legitimate' and
'Phishing'

f1 = [report[cls]['f1-score'] for cls in classes]

plt.figure(figsize=(8, 6))
bars = plt.bar(classes, f1, color='salmon', alpha=0.7)
plt.title("F1 Score by Class")
plt.ylabel("F1 Score")
plt.ylim(0, 1.05)
plt.axhline(y=np.mean(f1), color='red', linestyle='--',
label='Average F1 Score')
plt.legend()
for bar in bars:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() +
0.01, f"{bar.get_height():.2f}", ha='center')
plt.tight_layout()
plt.show()
```

FLASK SERVER

```
from flask import Flask, request, jsonify
import joblib
app = Flask(__name__)

ensemble_model =
joblib.load("phishing_ensemble_model.pkl")
vectorizer = joblib.load("tfidf_vectorizer.pkl")

@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.get_json()
        url = data["url"]

        features = vectorizer.transform([url])
        prediction = ensemble_model.predict(features)[0]
        confidence =
max(ensemble_model.predict_proba(features)[0])

        result = "phishing" if confidence >= 0.996 else "safe"
        return jsonify({"url": url, "prediction": result})

    except Exception as e:
        return jsonify({"error": str(e)})
if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=5000)
```

CHROME EXTENSION

A Chrome extension tool was developed to provide users with simple phishing web site identification. The extension uses HTML with CSS and JavaScript coding to create an easy-to-view minimal interface. The extension establishes communication with a machine learning model in the background through an API when you enter a URL. After one second passes the plugin provides a visual color

signal revealing if it identified the website as safe or unsafe.

background.js

```
chrome.webNavigation.onBeforeNavigate.addListener((details) => {
  const url = details.url;

  fetch("http://localhost:5000/predict", {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify({ url: url })
  })
    .then(response => response.json())
    .then(data => {
      if (data.prediction === "phishing") {
        alert("Phishing Website Detected!\nAccess Blocked");
        chrome.tabs.update(details.tabId, { url: "about:blank" }); // Block the website
      }
    })
    .catch(error => {
      console.error("API Error:", error);
    });
}, { urls: ["<all_urls>"] });
```

manifest.json

```
{
  "manifest_version": 3,
  "name": "Phishing Detect",
  "version": "1.0",
```

```
  "description": "Detects phishing websites in real-time",
  "permissions": ["tabs", "webRequest", "webNavigation", "activeTab", "scripting"],
  "host_permissions": ["http://*/*", "https://*/*"],
  "background": {
    "service_worker": "background.js"
  },
  "icons": {
    "128": "icon.png"
  },
  "action": {
    "default_popup": "popup.html",
    "default_icon": "icon.png"
  }
}
```

popup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Phishing Detector</title>
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="popup.js" defer></script>
  <style>
    @import
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600&display=sw
ap');
    body {
      background: linear-gradient(135deg, #4B0082, #8A2BE2); /* New purple-blue
```

```
gradient */
      font-family: 'Poppins', sans-serif;
    }
    .glass {
      background: rgba(255, 255, 255, 0.15);
      backdrop-filter: blur(10px);
      border-radius: 16px;
      padding: 20px;
      box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3);
    }
    h2 {
      font-family: 'Montserrat', sans-serif;
      font-weight: 600;
      letter-spacing: 0.8px;
    }
    input {
      border-radius: 25px; /* Smooth, curved input field */
      border: 2px solid rgba(255, 255, 255, 0.3);
      transition: all 0.3s ease;
    }
    input:focus {
      border-color: #ffffff;
      box-shadow: 0 0 8px rgba(255, 255, 255, 0.6);
    }
  </style>
</head>
<body class="w-80 h-52 flex flex-col justify-center items-center text-white">
  <div class="glass w-full text-center">
    <h2 class="text-xl flex items-center justify-center gap-2">
      <span class="text-white">Phishing Detector</span>
    </h2>
```

```
    <input type="text" id="urlInput" placeholder="Enter URL..."
      class="w-full px-4 py-2 mt-4 text-gray-900 bg-white focus:ring-2 focus:ring-purple-400 focus:outline-none transition-shadow shadow-md" />
    <div id="result" class="text-center mt-3 text-sm font-medium"></div>
  </div>
</body>
</html>
```

popup.js

```
document.getElementById("urlInput").addEventListener("input", function () {
  let url = this.value.trim();
  let resultElement = document.getElementById("result");

  if (url.length > 5) { // Prevent unnecessary API calls for short inputs
    fetch("http://localhost:5000/predict", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({ url: url })
    })
    .then(response => response.json())
    .then(data => {
      if (data.prediction === "phishing") {
        resultElement.innerHTML = "⚠️ <span class='text-red-400'>Phishing Website Detected!</span>";
      } else {
        resultElement.innerHTML = "☑ <span class='text-green-400'>Safe Website</span>";
      }
    })
```

```
    .catch(error => {
      console.error("API Error:", error);
      resultElement.innerHTML = "<span class='text-yellow-400'> ⚠️ Error Checking
URL</span>";
    });
  } else {
    resultElement.innerHTML = "";
  }
});
```

A2 – SCREENSHOTS

	A	B	C	D	E	F	G	H	I	J	K
1	URL	URLLength	Domain	DomainLengt	IsDomainIP	TLD	URLSimilar	CharContir	TLDLegitin	URLCharPr	TLDLength
2	https://www.southbankn	31	www.southb	24	0	com	100	1	0.522907	0.061933	3
3	https://www.uni-mainz.d	23	www.uni-ma	16	0	de	100	0.666667	0.03265	0.050207	2
4	https://www.voicefmrad	29	www.voicefn	22	0	uk	100	0.866667	0.028555	0.064129	2
5	https://www.sfnmjourna	26	www.sfnmjo	19	0	com	100	1	0.522907	0.057606	3
6	https://www.rewildingar	33	www.rewildi	26	0	org	100	1	0.079963	0.059441	3
7	https://www.globalrepor	30	www.globalr	23	0	org	100	1	0.079963	0.060614	3
8	https://www.saffronart.c	25	www.saffron	18	0	com	100	1	0.522907	0.063549	3
9	https://www.nerdscandy	25	www.nerdsca	18	0	com	100	1	0.522907	0.060486	3
10	https://www.hyderabadc	29	www.hyderal	22	0	in	100	1	0.005084	0.05698	2
11	https://www.aap.org	18	www.aap.org	11	0	org	100	1	0.079963	0.070497	3
12	https://www.religionenlil	33	www.religion	26	0	com	100	1	0.522907	0.063946	3
13	http://www.teramill.com	22	www.teramil	16	0	com	82.64463	1	0.522907	0.067418	3
14	https://www.socialpolicy	27	www.socialp	20	0	org	100	1	0.079963	0.064491	3
15	https://www.aoh61.com	20	www.aoh61.c	13	0	com	100	1	0.522907	0.055131	3

Fig A2.1 PhiUSIIL_Phishing_URL_Dataset

	A	B	C	D	E	F	G	H	I	J
1	url	length_url	length_hos	ip	nb_dots	nb_hyphen	nb_at	nb_qm	nb_and	nb_or
2	http://www.crestonwood.com	37	19	0	3	0	0	0	0	0
3	http://shadetreetechnology.c	77	23	1	1	0	0	0	0	0
4	https://support-appleld.com.s	126	50	1	4	1	0	1	2	0
5	http://rgipt.ac.in	18	11	0	2	0	0	0	0	0
6	http://www.iracing.com/track	55	15	0	2	2	0	0	0	0
7	http://appleid.apple.com-app.	32	24	0	3	1	0	0	0	0
8	http://www.mutuo.it	19	12	0	2	0	0	0	0	0
9	http://www.shadetreetechnol	81	27	1	2	0	0	0	0	0
10	http://vamoaestudiarmedicina	42	34	0	2	0	0	0	0	0
11	https://parade.com/425836/j	104	10	0	1	10	0	0	0	0
12	https://www.astrologyonline.c	56	22	0	3	0	0	0	0	0
13	https://www.lifewire.com/tcp	43	16	0	2	3	0	0	0	0
14	https://technofizi.net/top-bes	83	14	0	1	9	0	0	0	0
15	http://html.house/l7ceeid6.ht	31	10	0	2	0	0	0	0	0

Fig A2.2 Phishing _Dataset

```
PS D:\Jemina\Project\Dataset1> Invoke-RestMethod -Uri "http://localhost:5000/predict"
-Method Post -Headers @{"Content-Type"="application/json"} -Body '{"url": "http://secu
re-login.bankofamerica.online/verify"}'

confidence prediction url
---------- ---------- ---
         1 phishing   http://secure-login.bankofamerica.online/verify
```

Fig A2.3 Phishing Detection Prediction 1

```
PS D:\Jemina\Project\Dataset1> Invoke-RestMethod -Uri "http://localhost:5000/predict"
-Method Post -Headers @{"Content-Type"="application/json"} -Body '{"url": "https://www
.flipkart.com"}'

confidence prediction url
---------- ---------- ---
         0 safe       https://www.flipkart.com
```

Fig A2.4 Phishing Detection Prediction 2

```
PS D:\Jemina\Project\Dataset1> Invoke-RestMethod -Uri "http://localhost:5000/predict"
-Method Post -Headers @{"Content-Type"="application/json"} -Body '{"url": "http://payp
al.account-verification.com/login"}'

confidence prediction url
---------- ---------- ---
         1 phishing   http://paypal.account-verification.com/login
```

Fig A2.5 Phishing Detection Prediction 3

```
PS D:\Jemina\Project\Dataset1> Invoke-RestMethod -Uri "http://localhost:5000/predict"
-Method Post -Headers @{"Content-Type"="application/json"} -Body '{"url": "https://www
.amazon.com"}'

confidence prediction url
---------- ---------- ---
         0 safe       https://www.amazon.com
```

Fig A2.6 Phishing Detection Prediction 4

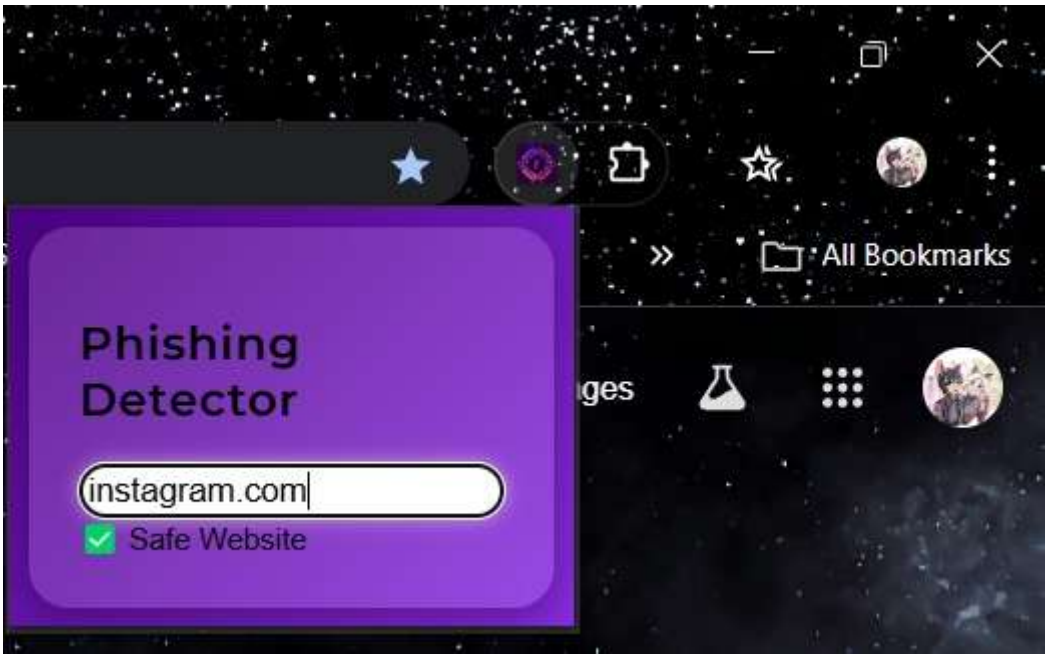

Fig A2.7 Chrome Extension prediction 1

Fig A2.8 Chrome Extension prediction 2

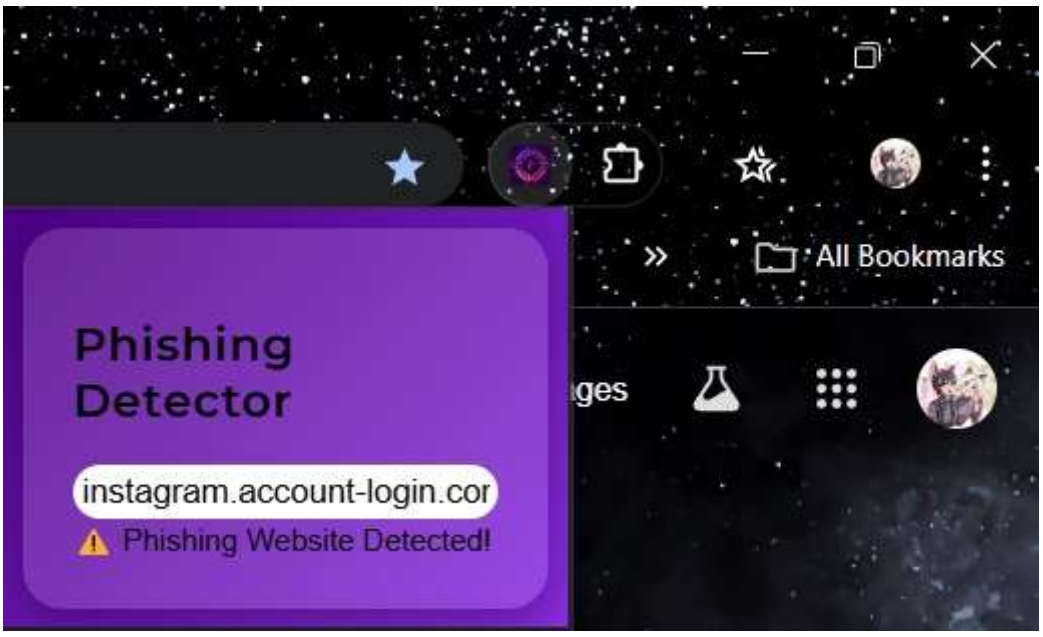

Fig A2.9 Chrome Extension prediction 3

Fig A2.10 Chrome Extension prediction 4

TECHNICAL BIOGRAPHY

Fathima Jemina M (210181601014), born in the year 2003 in Chennai Tamil Nadu, India. Completed my Intermediate in the year 2021. I am currently pursuing B. TECH Computer Science Engineering with specialization in Cybersecurity at the School of Computer Science and Engineering at B.S. Abdur Rahman Crescent Institute of Science and Technology which is located at Vandalur, Chennai, India. My interest lies in areas like Cyber security, Ethical hacking, vulnerability Assessment and Penetration Testing, Cyber Forensics. My email for communication fathijem1510@gmail.com and my contact number +91 9385495611.

Lakshmi Priya S (210181601022), born in the year 2004 in Chennai Tamil Nadu, India. Completed my Intermediate in the year 2021. I am currently pursuing B. TECH Computer Science Engineering with specialization in Cybersecurity at the School of Computer Science and Engineering at B.S. Abdur Rahman Crescent Institute of Science and Technology which is located at Vandalur, Chennai, India. My interest lies in areas like Cyber security, Ethical hacking, vulnerability Assessment and Penetration Testing, Cyber Forensics. My email for communication slakshmi.priya2202@gmail.com and my contact number +91 9884087125.