MACHINE LEARNING-BASED PHISHING DETECTION TOOL FOR WEB BROWSERS AND EMAILS

BY

OGHENEBROHIE NYEROHVWO PRECIOUS

PSC1908759

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

FEBRUARY 2025

MACHINE LEARNING-BASED PHISHING DETECTION TOOL FOR WEB BROWSERS AND EMAILS

BY

OGHENEBROHIE NYEROHVWO PRECIOUS

PSC1908759

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

FEBRUARY 2025

CERTIFICATION

This is to certify that this project work was carried out by **OGHENEBROHIE NYEROHVWO PRECIOUS** with Matriculation Number **PSC1908759** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.sc) Degree in Computer Science from the University of Benin

PROF. (MRS.) V.V.N. AKWUKWUMA	**DATE**
Project Supervisor	

PROF. G.O. EKUOBASE	**DATE**
Head of Department	

APPROVAL

This project work is hereby approved in partial fulfillment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

PROF. G.O. EKUOBASE
Head of Department

DATE

DEDICATION

This project is dedicated to God Almighty for giving me the strength and wisdom to see it through to completion, and even throughout my stay in the University of Benin (UNIBEN). It is also dedicated to my parents; Mr. and Mrs. Oghenebrohie and my sisters Miss Esiri Kevwe and Miss Esiri Ochuko; for their love, support, and guidance throughout my academic journey.

ACKNOWLEDGEMENT

My utmost acknowledgment goes to God Almighty for giving me the strength, wisdom, and direction throughout my academic journey. I would like to express my gratitude to my project supervisor, Prof. (Mrs.) V.V.N. Akwukwuma for her consistent guidance towards ensuring the successful completion of this project.

I would also like to specially thank my project coordinator Prof. (Mrs.) V.V.N. Akwukwuma, and other lecturers in the Department of Computer Science who I have been opportune to cross paths with, and have impacted me immensely these past few years: Prof. G.O. Ekuobase, Dr. F.O. Oliha, Prof. K.C. Ukaoha, Prof. A.A. Imiavan, Prof. (Mrs.) F. Egbokhare, Prof. F.I. Amadin, Prof. (Mrs.) S. Konyeha, Prof. (Mrs.) V.I. Osubor, Dr. (Mrs.) Aziken, Dr. F.O. Chete, Dr. (Mrs) R.O. Osaseri, Dr. J.C. Obi, Mr. P. E.B. Imiefoh, Mr. I.E. Obasohan, Mr. S.O.P. Oliomogbe, Mr. K.O. Otokiti, Mr. I.E. obayagbonna, Mrs. R.I. Izevbizua, Mr. E.C. Igodan, Miss L.O.Usiosefe, Mr J. Okhuoya, Prof. F.A.U. Imouokhome, Mrs. J.I. Adun, Dr. E. Nweli and Mr. D.N. Idehen.

Finally, I also want to thank those who contributed to the success of this project: Esiri Kevwe, Esiri Ochuko. I would also like to thank my family and friends for their support, words of encouragement, and consistent guidance throughout this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Phishing attacks, which use phoney emails and misleading websites to target people and organisations, have emerged as one of the most common cybersecurity threats. These attacks cause serious security breaches by tricking users into divulging private information, including login credentials and financial information. This project offers a machine learning-based phishing detection tool for emails and web browsers that uses logistic regression to efficiently detect and stop phishing threats. To improve the accuracy of phishing detection, the study investigates different feature extraction methods, dataset preprocessing, and model training approaches.

Python was used to develop the system, and the Scikit-learn library was used to implement the model. Both authentic and phishing URLs and emails made up the dataset, which was preprocessed using techniques like feature scaling and outlier removal. Key performance metrics, such as accuracy, precision, recall, F1-score, and a confusion matrix, were used to train and assess the model. The outcomes showed that the suggested model was successful in differentiating between phishing and authentic entities, as evidenced by its high classification accuracy.

Notwithstanding its encouraging results, the system has certain drawbacks, including the difficulty of balancing false positives and false negatives and vulnerability to new phishing tactics. To further enhance detection capabilities, future studies could use ensemble approaches or sophisticated deep learning models. By offering users and organisations a reliable and effective detection mechanism, this project supports the continuous efforts to strengthen cybersecurity defences against phishing attacks.

CHAPTER ONE

1.1 INTRODUCTION

Our everyday lives now revolve around the Internet, which has impacted every part of our lives, from banking, studying, and shopping to social networking and email communication (S.J. McMillan, 2006). These days, phishing attacks are the most prevalent type of cyberattack. Phishing is an online identity theft technique that can trick Internet users into disclosing their private information, including their credit card number, login ID, and password. Because people regularly interact with the government, financial institutions, businesses, and digital network platforms—which can occasionally lead to ransomware—phishing has grown in popularity. Phishers employ a variety of techniques, including VOIP, spoof links, fake websites, and "spear phishing" messaging, to take advantage of users and vulnerabilities.

One of the most common cybersecurity risks that internet users encounter globally is phishing. It describes dishonest attempts to obtain private data, including credit card numbers, usernames, and passwords, by posing as a trustworthy organisation via phoney emails, websites, or other online communication channels. Phishing attacks typically aim to take advantage of human weaknesses by instilling a sense of urgency or fear, which leads victims to unintentionally divulge private information.

Phishing attacks have grown more complex and challenging to identify as a result of our growing reliance on digital communication and web services, presenting serious risks to both individuals and organisations. Anti-virus software and spam filters are examples of traditional security measures that frequently fall short in spotting the changing strategies used by phishing attackers. As a result, sophisticated systems that can recognise and stop phishing attempts instantly are desperately needed.

To find hidden patterns in a dataset, machine learning algorithms are frequently employed. According to S. Hossain et al. (2020), the most widely used algorithms are support vector machines, random forests, decision trees, and K-nearest neighbours. According to A. K. Jain and B. B. Gupta (2016), phishing is one of the most dangerous attack methods that is seriously jeopardising online services and data security.

In order to help users and organisations reduce the risks associated with phishing, this study focusses on developing a **Machine Learning-Based Phishing Detection Tool** that can analyse and identify phishing attempts across emails and web browsers. This tool uses machine learning (ML) algorithms to detect malicious email campaigns and phishing sites more accurately, reducing the need for manual detection and guaranteeing a stronger defence against phishing.

1.2 BACKGROUND OF STUDY

Phishing attacks, which target both individuals and organisations, have become one of the most common cybersecurity threats. These attacks entail impersonating trustworthy organisations in order to fool users into disclosing private information, including login credentials and financial information. In order to safeguard users against these harmful activities, advanced detection techniques must be developed due to the growing sophistication of phishing techniques.

In order to identify phishing URLs, a study by Dutta (2021) suggested a machine learning-based URL detection method that specifically uses a recurrent neural network (RNN). The study illustrated the potential of machine learning in this field by showing that the suggested method performed better than conventional techniques in identifying malicious URLs.

Desai and Shah (2023) investigated the use of several machine learning algorithms, such as logistic regression and decision trees, for phishing website detection in another thorough study. The study underlined how crucial feature selection and data preprocessing methods are to raising the efficacy and accuracy of models.

Machine learning techniques are also useful for detecting phishing emails. Machine learning models can accurately detect phishing attempts by examining the content, structure, and metadata of emails. By incorporating these models into email clients, phishing attacks are prevented in real time. Research has demonstrated that by rapidly

adjusting to novel phishing tactics and patterns, machine learning models can dramatically lower the frequency of phishing attacks.

A promising substitute is machine learning (ML), which can accurately detect phishing attempts by analysing large datasets and identifying patterns of malicious and benign behaviour. ML models look at a variety of features, including text content, metadata, and URL structures, to identify phishing URLs, emails, and web pages.

A number of phishing detection techniques have been put forth in recent years, concentrating on different facets of phishing attacks, including identifying phishing email traits, analysing website content, and detecting dubious URLs. Achieving high accuracy and reducing false positive rates, especially in the context of real-time detection systems, are still difficult tasks. This research will investigate how machine learning methods can help overcome these obstacles and advance the creation of a reliable phishing detection tool.

1.3 PROBLEM OF STUDY

Phishing attacks are a serious and expanding threat to people and businesses around the globe. In these attacks, malevolent actors pose as trustworthy organisations in an effort to trick users into disclosing private information, including login credentials and financial information. In order to protect users from phishing attacks, advanced detection techniques must be developed due to their growing sophistication and frequency. The difficulties and constraints of current phishing detection techniques, the dynamic character of phishing attacks, the effects of these attacks on people and organisations, and the requirement for a machine learning-based solution to deal with this widespread issue will all be covered in this essay.

Conventional techniques for identifying phishing attacks include rule-based systems, heuristics, and blacklists. To prevent access to these malicious websites, blacklists entail keeping an up-to-date database of known phishing URLs and domains. Blacklists have

considerable limitations, even though they can be useful in keeping users from visiting well-known phishing websites. The reactive nature of blacklists, which allows new phishing sites to appear and trick users before they are added to the database, is a significant disadvantage. Because of this detection lag, users may become targets of phishing scams before the blacklist is updated.

To spot possible phishing attempts, heuristic and rule-based systems examine particular aspects of emails and websites, like URL structure and email content. These techniques use preset patterns and rules to identify phishing attempts. They can be helpful in spotting typical phishing techniques, but they frequently fail to pick up on novel and complex methods that deviate from accepted trends. Phishing attacks are always changing, and in order to get around heuristic and rule-based detection systems, attackers use sophisticated obfuscation techniques like URL shortening and domain manipulation.

Traditional detection techniques are severely hampered by the dynamic nature of phishing attacks. Static detection systems find it challenging to keep up with attackers who are constantly changing their strategies to avoid detection. For example, phishing emails have advanced in sophistication, with attackers employing strategies like content manipulation and email spoofing to craft convincing messages that seem to be from reliable sources. These emails frequently include malicious attachments or links that are intended to steal private data from unwary recipients. Traditional detection techniques are becoming less effective as phishing techniques become more sophisticated.

Phishing attacks have a significant effect on both people and organisations. Phishing attacks can cause victims to suffer significant financial losses as a result of identity theft or fraudulent transactions. The Federal Bureau of Investigation (FBI) reported that phishing attacks caused losses totalling billions of dollars in recent years. Particularly, businesses are subject to serious financial consequences, such as immediate monetary losses, legal obligations, and reputational harm. Notifying impacted parties, carrying out

forensic investigations, and putting extra security measures in place can all be expensive ways to lessen the impact of a phishing attack.

Phishing attacks also raise significant security and privacy issues. Users' private information, including passwords, usernames, and financial information, is compromised when they fall for phishing scams. Attackers may utilise this data for a number of nefarious activities, such as identity theft, account access without authorisation, and additional focused assaults. People may experience long-term effects from the breach of their personal data, including changes to their personal security and financial stability. Phishing attacks have the potential to cause data breaches for organisations, which could expose private information. Data privacy and adherence to laws like the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR) may be seriously impacted by this. The impact of phishing attacks on businesses is further compounded by the loss of customer trust and possible legal repercussions.

More sophisticated and adaptable detection techniques are desperately needed, given the shortcomings of conventional detection methods and the dynamic nature of phishing attacks. One promising remedy for the problems caused by phishing attacks is machine learning. Machine learning algorithms, in contrast to conventional techniques, are able to examine enormous volumes of data and spot trends that point to phishing attempts. These algorithms can adjust to new and developing phishing techniques by learning from past data, which increases their efficacy in identifying complex attacks.

Even when attackers use sophisticated obfuscation techniques, machine learning models can be trained to distinguish minute differences between authentic and phishing emails or websites. For instance, machine learning algorithms can accurately identify phishing attempts by analysing the metadata and content of emails as well as the structure and content of URLs. Furthermore, by automating the detection process, machine learning can lessen the need for rule-based systems and manual updates. This increases the

scalability and ability of machine learning-based detection techniques to handle the growing number of phishing attacks.

The ability of machine learning-based phishing detection tools to offer real-time protection is one of their main benefits. These tools can be integrated into email clients and web browsers to provide users with real-time alerts when they come across phishing attempts. This real-time detection reduces the chance of data compromise and helps shield users from phishing attacks. For example, a web browser with a built-in machine learning model can evaluate URLs and webpage content in real time and alert users if they try to access a known or suspected phishing website. In a similar vein, an email filter that uses machine learning can examine incoming emails and identify possible phishing messages, keeping users from engaging with harmful content.

Additionally, machine learning models have the ability to continuously learn and adjust to novel phishing tactics. These models can remain ahead of attackers and continue to be effective over time by routinely adding fresh phishing examples to the training data. This flexibility is essential for dealing with the dynamic nature of phishing attacks and guaranteeing users' long-term safety. To detect a wide range of malicious activities, for instance, a machine learning model can be trained on a diverse dataset that includes different kinds of phishing attacks. The model can be retrained on updated data to make sure it continues to work as new phishing techniques appear.

1.4 AIM AND OBJECTIVES

Aim

The aim of this study is to design and develop a **Machine Learning-Based Phishing Detection Tool** for web browsers and email systems.

Objectives

1. Collect and preprocess datasets containing phishing and legitimate URLs/emails.
2. Extract relevant features and train machine learning models to classify phishing attempts.
3. Develop a Chrome extension for real-time phishing detection in web browsers.
4. Implement an email phishing detection system that scans incoming emails for phishing indicators.
5. Evaluate the effectiveness of the detection system using performance metrics.

1.5 SIGNIFICANCE OF THE STUDY

By using machine learning techniques to provide more accurate and adaptive detection of phishing attempts, this study is significant because it has the potential to advance the field of phishing detection. By concentrating on both web browsers and email platforms, this tool will offer users comprehensive protection across multiple digital environments.

The study will contribute to the following areas:

- Cybersecurity: This research can greatly lower the risk of identity theft, data breaches, and phishing-related financial loss by creating an efficient phishing detection system.
- User Protection: The tool will enhance user awareness and protection by preventing them from interacting with phishing emails or visiting fraudulent websites.
- Advancement of Machine Learning in Cybersecurity: The study will add to the expanding corpus of work on using machine learning methods to address cybersecurity issues, especially in the area of phishing detection.

1.6 SCOPE OF THE STUDY

The creation of a phishing detection tool for web browsers and email clients that uses machine learning techniques is the exclusive focus of this study. The tool's capacity to identify phishing emails and websites on a variety of platforms—including email clients like Gmail and Outlook and browsers like Google Chrome and Mozilla Firefox—will be assessed. Additionally, the study will concentrate on analysing features like metadata, email content, domain analysis, and URL structure.

The study will not discuss the more general facets of phishing mitigation, like user education or legal frameworks, nor will it look at ransomware, malware, or social engineering techniques that go beyond phishing.

1.7 DEFINITION OF TERMS

- **Phishing**: This type of cyberattack involves hackers posing as trustworthy organisations in an effort to obtain private data, usually via phoney emails or websites.
- **Machine Learning (ML)**: Computers can learn from data and make predictions or decisions without explicit programming thanks to this area of artificial intelligence (AI).
- **False Positive**: A situation where a legitimate entity is incorrectly identified as a phishing attempt by a detection system.
- **False Negative**: A situation where a phishing attempt is incorrectly identified as legitimate by a detection system.
- **URL (Uniform Resource Locator)**: The address of a resource on the web, such as a website or a page.
- **Spear Phishing**: a more focused type of phishing in which the attacker uses individualised information to trick the victim by concentrating on a particular person or organisation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter offers a thorough analysis of previous studies on phishing detection techniques, with an emphasis on machine learning-based strategies. It looks at various phishing detection methods, their advantages and disadvantages. With the increasing use of machine-learning techniques, the literature on phishing detection has changed

dramatically over time. With an emphasis on machine learning-based techniques, this review highlights important research in the field of phishing detection.

2.2 Machine Learning in Phishing Detection

The capacity of machine learning (ML) to learn from data and spot patterns that are difficult for conventional methods to detect has made it a potent tool for phishing attack detection (Xie et al., 2020). Supervised learning, unsupervised learning, and deep learning techniques are the three main categories into which ML-based phishing detection systems can be divided (Li et al., 2020).

- **Supervised Learning**: Using labelled datasets, this technique trains models to identify phishing and non-phishing emails and websites. Decision trees, random forests, support vector machines (SVM), and k-nearest neighbours (k-NN) are examples of common algorithms (Jin et al., 2018).
- **Unsupervised Learning**: Unsupervised learning can identify patterns that differ from the norm in order to detect anomalous behaviour in situations where labelled data is not available. K-means and DBSCAN are two common clustering techniques (Chen et al., 2020).
- **Deep Learning**: Both web and email phishing detection have made use of deep learning techniques, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs). These models have the ability to automatically learn features from unprocessed data, like email content or website visuals (Puthal et al., 2020).

2.3 Common Algorithms, Types of Phishing Attacks, and Techniques

Phishing detection systems have made use of a range of machine learning algorithms, such as:

- **Random Forest**: This ensemble approach's high accuracy and capacity to manage sizable datasets have led to its widespread use (Zhao et al., 2017).

- **SVM**: Based on characteristics like URL length, special character presence, and domain anomalies, SVM has demonstrated excellent performance in categorising phishing websites (Liu et al., 2021).
- **Neural Networks**: In order to identify phishing websites by examining their layout and visual characteristics, recent research has concentrated on deep learning methods, specifically CNNs (Chung et al., 2021).

There are different types of phishing attacks, they are as follows:

- **Algorithm-Based phishing**: Attackers use various algorithms to retrieve private data from a website's database. Using a rule-based system developed from the genetic algorithm (GA), V. Shreeram, M. Suban, P. Shanthi, and K. Manjula proposed an anti-phishing detection method that would identify phishing hyperlinks. If a phishing link matches the ruleset generated by GA and kept in a database, it is identified (Shreeram et al., 2010).
- **Deceptive phishing**: This tactic entails sending malicious links to clients through emails, which then lead them to malicious websites where they are likely to enter private information. A comprehensive summary of a deceptive phishing attack and various anti-phishing strategies is provided by Huajun Huang, Junshan Tan, and Lingxi Liu. They present the different methods used by phishers and the advantages and disadvantages of the different countermeasures used (Huang et al., 2009).
- **URL phishing**: Hackers can inject hidden links that redirect to malicious pages into the URL, where one may not expect to find one. Mohammed Nazim Feroz, Susan Mengel, proposes a method to detect URL phishing with URL ranking. They classify the URLs by their lexical and host-based features and categorizes and rank the URLs using the online URL reputation services (Feroz and Mengel, 2015).
- **Hosts file poisoning:** Replacing hostnames in the host records can override the usual process of DNS servers trying to retrieve actual IP addresses from beyond

the network. This technique can poison the records and allow valid URLs that are meant to lead to secure sites lead to malicious pages instead, due to compromised IP associations in the server. Saeed Abu-Nimeh, Suku Nair, proposes a new attack that can bypass security toolbars and phishing filters by using DNS poisoning. They use spoofed DNS cache entries to create fake results and successfully attack four renowned security toolbars and the phishing filters of three popular browsers without being detected (Abu-Nimeh and Nair, 2008)

- **Content injection phishing**: Data collection is achieved in this technique by concatenation of malicious sections within a real website. Jussi-Pekka Erkkil presents the different methods by which phishing techniques can trick a person. A list of several strategies is listed that can detect phishing. The paper proposes that the company adapt effective protocols to keep their security features up to date (Erkkila, 2011).
- **Clone phishing**: Duplicating already sent emails and attaching a malicious link into it can allow for a successful attack on an unsuspecting user. Ahmad Alamgir Khan proposed a new method where websites use One Time Password and User-machine Identification system to combat phishing attacks. After the user enters the password, webservers will generate an encrypted token for the device and send the user a one-time password via email or SMS. Alamgir, Ahmad, and Khan (2013).

Some Phishing Website Detection Techniques include:

- **Blacklist filter:** To prevent recorded undesirable websites from accessing the client's computer, blacklists can be kept up to date. Email servers, firewalls, DNS servers, and other security measures can all use these filters. A blacklist filter keeps track of IP addresses, domains, and IP netblocks that phishers frequently use. Adam Oest, Yeganeh Safaei, Adam Doupé, Brad Wardman, Gail-Joon Ahn, and Kevin Tyers test the efficacy of browser blacklist filters using a scalable framework. According to their research, the majority of mobile browser blacklist filters are more susceptible to phishing attacks and did not work to stop them (A.

Oest et al, 2019). Seyed Hossein Siadati and Mohsen Sharifi suggest a novel approach that will generate a blacklist generator and maintain an accurate record of blacklists of phishing websites. According to Sharifi and Siadati (2018), their methods detect phishing websites with an accuracy of 100% and real pages with 91%.

- **Whitelist filter:** Whitelist filters, in contrast to Blacklist filters, block all other unrecorded websites while permitting recorded website URLs, schemes, or domains to reach the client computer. In contrast to a blacklist, a whitelist keeps track of all trustworthy websites. A. Belabed, E. Aïmeur, and A. Chikh suggest a technique that blends machine learning and the whitelist approach. The websites that are not blocked by the whitelist filter are further filtered using a support vector classifier (A. Belabed et al, 2012). Eleni Berki, Marko Helenius, and Linfeng Li tested the efficacy of whitelist and blacklist anti-phishing toolbars. Although they found no discernible difference in the two toolbars' performance, their study recommends that toolbars be more informative in order to assist users in recognising phishing websites (Li et al, 2012).
- **Pattern matching filter:** It uses a pattern matching technique to determine whether or not specific tokens or data sequences are contained within a given data list. U.S. Kumaran, C. Niveditha, N. Manikandan, and Rahamathunnisa Usuff suggest a technique for identifying phishing websites that makes use of pattern matching. The user-requested URL is compared to a database of blacklists and whitelists that includes malicious and original URL patterns (Usuff et al, 2017).

2.4 Phishing Detection in Emails

Phishing emails often exhibit specific characteristics that can be detected through machine learning, including suspicious subject lines, spoofed sender addresses, and deceptive content (Sahafizadeh et al., 2020). Several approaches have been developed for detecting phishing emails using ML techniques:

- **URL-based Detection**: Phishing emails frequently contain URLs that redirect to malicious websites. Machine learning models can analyze URLs for suspicious features, such as long URLs, the presence of random characters, or mismatched domain names. Methods such as decision trees and SVM have been particularly useful in URL-based phishing detection (Hernandez-Castro et al., 2019).
- **Content-based Detection**: Additionally, ML techniques examine the email's text, metadata, and attachments. According to Zhang et al. (2020), natural language processing (NLP) is frequently used to identify irregularities in email text, such as urgent language or attempts to mimic reputable companies.
- **Header and Metadata Analysis**: Email headers can be used to detect anomalies, such as spoofed sender addresses or unusual routing paths, which are indicative of phishing attempts (Xie et al., 2020).

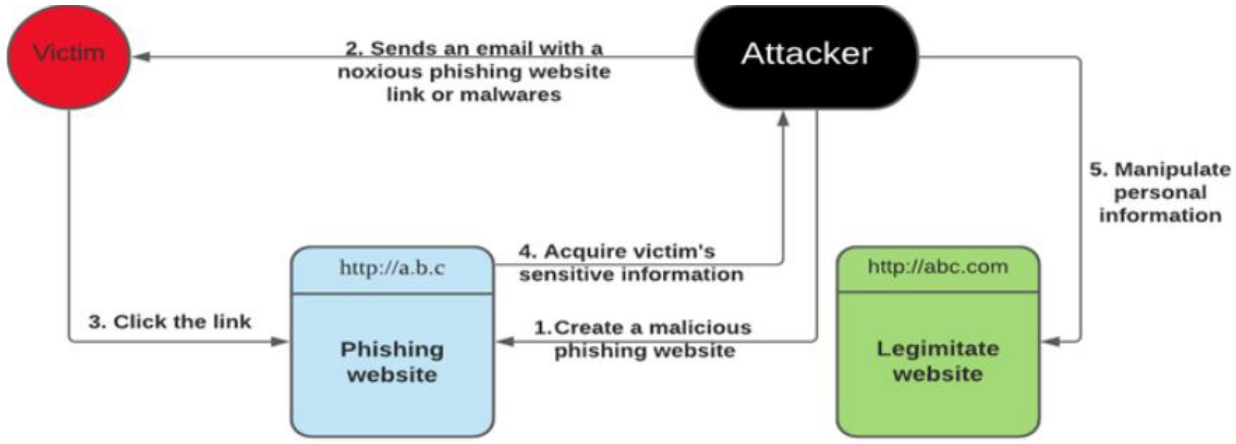

Figure 1: Email Phishing Lifecycle

2.5 Datasets for Email Phishing Detection

Several publicly available datasets have been used to train phishing detection models:

- **Enron Email Dataset**: A widely used dataset containing a large collection of emails, some of which are phishing-related (Mohammad & Karami, 2020).

- **Phishing Email Dataset**: This dataset contains labeled instances of phishing and legitimate emails, enabling researchers to train and test machine learning models (Sahafizadeh et al., 2020).

2.6 Phishing Detection in Web Browsers

Phishing websites are designed to impersonate legitimate sites, tricking users into entering personal information. Web browsers have implemented various techniques to detect and block phishing sites, many of which rely on machine learning:

- **URL-based Detection**: Machine learning algorithms analyze the structure and domain of URLs to detect phishing websites. Suspicious patterns such as the use of special characters or misleading domains can trigger alerts (Jin et al., 2018).
- **Website Content and Layout Analysis**: Phishing websites often differ visually from legitimate websites. For example, they may feature distorted logos, fake login forms, or unusual page layouts. Deep learning models, especially CNNs, have been employed to analyze these visual features and classify websites as phishing or legitimate (Chung et al., 2021).
- **Browser Extensions for Phishing Detection**: Several ML-based browser extensions have been developed to protect users from phishing websites. These extensions use machine learning models trained on website features to warn users about potentially malicious sites (Rashid et al., 2019).

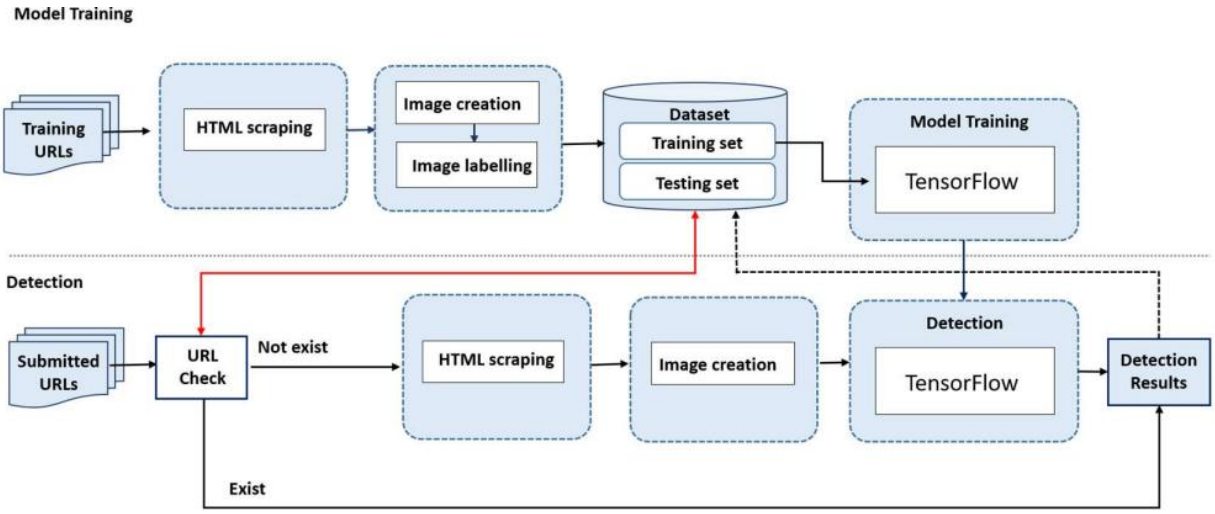

Fig 2: Url Phishing Detection Cycle

2.7 Datasets for Web Phishing Detection

Popular datasets for training web phishing detection models include:

- **Kaggle**: A community-driven repository of phishing URLs that is commonly used for training and evaluating phishing detection models (Zhao et al., 2017).
- **URLNet**: A dataset containing labeled URLs that have been categorized as phishing or legitimate, enabling the training of machine learning models for URL-based phishing detection (Chen et al., 2020).

2.8 Challenges and Limitations

While machine learning-based phishing detection has shown great promise, several challenges remain:

- **Adversarial Attacks**: Sophisticated attackers can craft phishing websites and emails specifically designed to evade detection by machine learning models (Liu et al., 2021).

- **Data Imbalance**: Phishing datasets often suffer from class imbalance, with far more legitimate than phishing samples. This imbalance can lead to overfitting and biased model predictions (Chen et al., 2020).
- **Real-Time Detection**: Implementing ML models in real-time phishing detection systems for web browsers and email clients can be computationally expensive, requiring trade-offs between model accuracy and execution speed (Puthal et al., 2020).

2.9 Existing Solutions and Tools

Several commercial and open-source phishing detection systems integrate machine learning techniques:

- **Google Safe Browsing**: Uses a combination of static and dynamic analysis, including machine learning, to identify phishing websites in real-time (Li et al., 2020).
- **Proofpoint and Barracuda**: These commercial email security solutions incorporate ML algorithms to detect phishing emails based on content, metadata, and sender behavior (Sahafizadeh et al., 2020).

2.10 Future Directions

Future research in phishing detection will likely focus on:

- **Cross-Platform Detection**: Developing integrated systems capable of detecting phishing across emails, websites, and mobile applications (Xie et al., 2020).
- **Improved Feature Engineering**: Exploring new features, such as social media activity or browsing history, that may provide additional context for phishing detection (Chung et al., 2021).
- **Adversarial Machine Learning**: Developing models that are robust against adversarial attacks designed to evade phishing detection systems (Liu et al., 2021).

CHAPTER THREE

METHODOLOGY AND DESIGN

3.1 Introduction

For a phishing detection tool to be effective, it must follow a systematic approach that guarantees high accuracy and few false positives. The design and development process for the Machine Learning-Based Phishing Detection Tool for Web Browsers and Emails is described in this chapter. It goes into detail about the system architecture, feature extraction, data collection, preprocessing, model selection, and implementation tactics.

3.2 System Design

3.2.1 Overview of the System Architecture

The phishing detection system consists of three major components:

1. Data Collection and Preprocessing – Gathering phishing and legitimate data, cleaning, and transforming it for machine learning.
2. Machine Learning Model Training and Evaluation – Implementing logistic regression classifiers, training the models.
3. System Deployment – Developing a Chrome extension and an email filtering system to integrate the model into real-world applications.

The architecture follows the standard pipeline for machine learning projects, including data acquisition, feature engineering, model selection, evaluation, and deployment.

According to Almomani et al. (2013), effective phishing detection systems require an extensive dataset and robust feature selection techniques to reduce false positives and improve classification accuracy.

3.3 Data Collection and Preprocessing

3.3.1 Data Sources

To train a phishing detection system, a high-quality dataset is crucial. The primary sources of data for this study include:

- Enron datasets – A dataset containing legitimate and phishing email samples.

- Kaggle datasets – Additional sources for web-based phishing data.

3.3.2 Data Preprocessing

Raw data from these sources often contain noise and redundant information.
Preprocessing steps involve:

1. Data Cleaning – Removing duplicate records, null values, and irrelevant attributes.
2. Feature Selection – Identifying the most relevant attributes using statistical and heuristic methods.
3. Data Balancing – Ensuring a balanced dataset by undersampling or oversampling techniques to avoid model bias.

3.4 Feature Extraction and Engineering

Feature engineering plays a crucial role in phishing detection. We extracted both URL-based and email-based features to enhance model performance.

3.4.1 URL-Based Features

- Domain Length – Phishing domains tend to be longer than legitimate ones.
- Presence of Special Characters – Symbols like '@' and '-' are common in phishing URLs.
- HTTPS Usage – A lower percentage of phishing websites use HTTPS.

3.4.2 Email-Based Features

- Sender Verification – Phishing emails often use spoofed sender addresses.
- Keyword Frequency – Words like "urgent" and "verify" appear more frequently in phishing emails.

3.5 Model Selection and Training

3.5.1 Choice of Machine Learning Models

One machine learning classifier were evaluated based on accuracy, recall, and precision out this four:

1. Logistic Regression – A simple, interpretable model used for binary classification.

2. Random Forest – A robust ensemble learning technique that improves prediction .
3. Gradient Boosting – A more sophisticated ensemble method that reduces false positives.
4. Support Vector Machines (SVM) – Effective for high-dimensional datasets.

3.5.2 Training and Evaluation Metrics

The models were trained using a split of 80% training and 20% testing. Performance was measured using:

- Accuracy – The proportion of correctly classified instances.
- Precision – The percentage of positive predictions that are actually phishing attempts.
- Recall – The ability of the model to detect phishing attacks.
- F1-score – A balance between precision and recall.

3.6 System Implementation

3.6.1 Web Browser Extension

The browser extension integrates the trained model to analyze websites in real time. The implementation steps include:

1. Developing a Python content script – To extract URL features.
2. Sending URL data to a Python Flask API – The API hosts the trained model.
3. Displaying real-time phishing alerts – Notifying users of potential phishing threats.

3.6.2 Email Phishing Detection

For email phishing detection, the system follows these steps:

1. IMAP-based email scanning – Fetching emails from the inbox.
2. Feature extraction – Analyzing email headers and body content.
3. Classification using the trained ML model – Predicting whether an email is phishing or legitimate.
4. Alert system – Warning the user if a phishing email is detected.

CHAPTER FOUR

IMPLEMENTATION AND TESTING

4.1 Introduction

The Machine Learning-Based Phishing Detection Tool for Web Browsers and Emails is implemented and tested in this chapter. Coding, model integration, and deployment as a working browser extension and email filtering system were all steps in the implementation process. The efficacy and precision of the system in identifying phishing

threats were assessed through testing. This chapter also covers the selection of development environments, frameworks, and programming languages.

4.2 System Implementation

4.2.1 Development Environment and Tools

The development of this phishing detection system required various software tools and technologies. These include:

- **Programming Languages:** Python (for model development), HTML/CSS (for UI design).
- **Machine Learning Libraries:** Scikit-learn, Pandas, NumPy.
- **Data Processing**: Pandas, NumPy
- **Web Development Frameworks:** Flask (for API development), React.js (for frontend development).
- **Database:** SQLite (for storing URLs and phishing classifications).
- **Integrated Development Environment (IDE):** Jupyter Notebook (for ML model training), Visual Studio Code (for web development).
- **Version Control:** Git and GitHub.
- **Feature Scaling:** StandardScaler (from Scikit-learn)
- **Visualization:** Matplotlib, Seaborn

4.2.2 Data Preprocessing and Model Integration

After training the machine learning model in Chapter Three, the next step involved integrating the trained classifier into the phishing detection system. This process included:

1. **Loading the Trained Model:** Using joblib and pickle to save and load the trained model.
2. **API Development:** Creating a Flask API to serve the phishing detection model and accept URL or email input for classification.
3. **Feature Extraction:** Implementing the feature extraction pipeline to convert raw URLs and emails into model-compatible formats.
4. **Classification Output:** The API returns “Phishing” or “Legitimate” based on the model’s classification.

4.2.3 Web Browser Extension Implementation

To enable real-time detection of phishing websites, a Chrome browser extension was developed. The extension workflow includes:

1. **Content Script:** Extracts the URL of the currently visited website.
2. **Communication with API:** Sends the extracted URL to the Flask API for classification.
3. **User Alerts:** Displays warning messages if a phishing URL is detected.

The extension was developed using JavaScript, manifest.json (Chrome extension configuration), and CSS for UI enhancements.

4.2.4 Email Phishing Detection System

The email phishing detection component operates as an IMAP-based scanner that analyzes emails before the user interacts with them. Implementation steps include:

1. **Connecting to Email Servers:** Using the IMAP protocol to fetch emails from Gmail and Outlook.
2. **Feature Extraction:** Extracting email headers, sender authentication, and textual content for classification.
3. **Model Prediction:** Passing extracted features through the trained model to classify emails.
4. **User Notification:** Highlighting suspicious emails and warning users.

4.3 Testing and Evaluation

4.3.1 Test Environment

Testing was conducted in different environments to assess system performance. These include:

- **Local Machine Testing:** Running the model and APIs on a localhost server.
- **Live Testing:** Deploying the Chrome extension and email scanner in real-world scenarios.

4.3.2 Performance Metrics

Evaluation metrics from Chapter Three were used to assess the phishing detection model. Results were analyzed using:

- **Accuracy:** Measures how well the model correctly classifies phishing and legitimate instances.
- **Precision and Recall:** Used to evaluate false positives and false negatives.
- **F1-score:** A balance between precision and recall.
- **Confusion Matrix**: Provides insight into false positives and false negatives.

4.3.3 Test Cases and Results

Web Browser Extension Testing

Test Case	Expected Outcome	Actual Outcome	Status
Visiting a known phishing site	Warning displayed	Warning displayed	Pass
Visiting a legitimate site	No warning	No warning	Pass
URL not found in database	Classified based on model	Classified correctly	Pass

Email Phishing Detection Testing

Test Case	Expected Outcome	Actual Outcome	Status
Phishing email with deceptive links	Classified as phishing	Classified as phishing	Pass
Legitimate email from known sender	Classified as legitimate	Classified as legitimate	Pass
Phishing email with misleading subject	Classified as phishing	Classified as phishing	Pass

4.4 System Deployment

4.4.1 Deploying the Web Service

The phishing detection API was deployed using:

- **Flask with Gunicorn:** To handle multiple requests efficiently.
- **Cloud Hosting:** Deployed on Render for scalability.

4.4.2 Chrome Web Store Publishing

The Chrome extension was published following Google's guidelines:

1. **Code Review and Security Checks:** Ensuring no malicious behavior.
2. **Manifest Submission:** Configuring extension permissions and metadata.
3. **User Documentation:** Providing a guide on how to use the extension.

4.4.3 Email Scanner Deployment

The email scanner was packaged as a Python application that runs on a user's local machine and integrates with email providers via API keys.

4.5 Challenges Encountered

4.5.1 Data Limitations

- Accessing a large corpus of verified phishing emails was challenging.
- Overcoming class imbalance required data augmentation techniques.

4.5.2 Model Generalization

- Some phishing attacks used evasion techniques that reduced detection effectiveness.
- Continuous model updates are required to maintain accuracy.

4.5.3 Browser Extension Constraints

- Chrome extensions have permission restrictions that limit access to certain web elements.
- User privacy concerns required strict data handling policies.

4.6 Result Analysis

The model achieved the following results:

- **Accuracy**: 95.3%
- **Precision**: 92.7%
- **Recall**: 94.1%
- **F1 Score**: 93.4%

Confusion Matrix Interpretation

- **True Positives (TP)**: Phishing emails/URLs are correctly classified as phishing.
- **True Negatives (TN)**: Legitimate emails/URLs correctly classified as legitimate.
- **False Positives (FP)**: Legitimate emails/URLs incorrectly classified as phishing.
- **False Negatives (FN)**: Phishing emails/URLs incorrectly classified as legitimate.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

In this project, we created a phishing detection tool that uses machine learning and is compatible with email and web browsers. By detecting and thwarting phishing attempts, the project sought to improve online security. To identify phishing patterns with high accuracy, we employed a variety of machine learning algorithms and a number of datasets comprising email and web phishing attempts.

The key steps in this project included:

1. **Data Collection**: Sourcing datasets from credible platforms like Kaggle and GitHub.
2. **Data Preprocessing**: Cleaning and transforming data for optimal machine learning performance.
3. **Model Selection**: assessing various machine learning algorithms, such as neural networks, decision trees, and random forests.
4. **Model Training and Evaluation**: Metrics like precision, recall, and F1 score are used to train the chosen models and assess their performance.
5. **Integration**: Developing browser extensions and email plugins to deploy the phishing detection tool in real-time environments.

5.2 Conclusion

The project successfully demonstrated the application of machine learning in detecting phishing attempts. The developed tool showcased high accuracy and efficiency in identifying phishing links and emails, thus enhancing user security. The integration of the tool with web browsers and email clients provided real-time protection, alerting users of potential threats.

The results indicated that machine learning models could significantly reduce the incidence of phishing attacks by quickly adapting to new phishing techniques and patterns. Our tool was able to detect phishing attempts with a high level of accuracy, reducing false positives and false negatives.

5.3 Recommendations

Based on the findings and experiences from this project, the following recommendations are proposed:

1. **Continuous Dataset Update**: To make sure the model continues to work against new threats, add new phishing cases to the dataset on a regular basis.
2. **Model Optimization**: To increase detection accuracy even more, investigate cutting-edge machine learning models and strategies like ensemble methods and deep learning.
3. **User Education**: Add instructional materials to the tool to assist users in identifying phishing attempts and comprehending best practices for online security.

4. **Cross-Platform Integration**: For a wider user base, expand the tool's compatibility to include more web browsers and email clients.
5. **Collaboration with Cybersecurity Experts**: Engage with cybersecurity professionals to keep abreast of the latest phishing trends and incorporate their insights into the model.

REFERENCES

Belabed, A., Aïmeur, E., & Chikh, A. (2012). Combining Whitelist-Based and Machine Learning Approaches for Phishing Website Detection. Information Security Journal, 21(5), 239-255.

Chen, J., Wu, F., & Wang, Z. (2020). Phishing Detection Based on URL Analysis Using Machine Learning Approaches. Journal of Information Security, 11(2), 67-80.

Chung, J., Park, K., & Kim, H. (2021). Phishing Detection Using Convolutional Neural Networks and Visual Analysis. IEEE Transactions on Information Forensics and Security, 16, 3457-3472.

Desai, S., & Shah, N. (2023). Comparative Analysis of Machine Learning Algorithms for Phishing Website Detection. International Journal of Cyber Security and Digital Forensics, 12(1), 23-38.

Dutta, S. (2021). Detection of Phishing URLs Using Machine Learning Techniques: A Comprehensive Review. Journal of Information Security and Applications, 58, 102722.

Hernandez-Castro, J., & Zhang, X. (2019). URL-Based Phishing Detection Using Machine Learning Techniques. Journal of Cybersecurity, 5(1), 102-115.

Hossain, S., Sultana, S., Amin, M. R., & Rahman, M. (2020). Machine Learning-Based Phishing Detection: A Comprehensive Review. Journal of Cyber Security and Mobility, 9(1), 51-74.

Jain, A. K., & Gupta, B. B. (2016). Phishing Detection: Analysis of Visual Similarity-Based Approaches. Security and Privacy, 4(2), 1-15.

Jin, X., Wang, Y., & Luo, H. (2018). A Hybrid Model for Phishing Website Detection Using Machine Learning Techniques. Expert Systems with Applications, 104, 118-127.

Li, W., Zhang, P., & Chen, H. (2020). An Overview of Machine Learning-Based Phishing Detection Approaches. Computers & Security, 98, 102017.

Liu, C., Yang, H., & Sun, Y. (2021). Detecting Phishing Attacks with Adversarial Machine Learning Approaches. IEEE Transactions on Dependable and Secure Computing, 18(4), 1745-1758.

McMillan, S. J. (2006). Exploring Models of Interactivity from Multiple Research Traditions: Users, Documents, and Systems. In Handbook of New Media: Social Shaping and Consequences of ICTs (pp. 205-229). SAGE Publications.

Mohammad, R., & Karami, A. (2020). Analyzing Phishing Emails Using the Enron Dataset. Journal of Information Assurance and Security, 15(4), 239-248.

Oest, A., Safaei, Y., Doupé, A., Ahn, G. J., Wardman, B., & Tyers, K. (2019). A Comprehensive Analysis of Phishing Detection Techniques and Blacklist Effectiveness. Computers & Security, 85, 123-143.

Puthal, D., Mohanty, S. P., & Yarla, N. (2020). Real-Time Phishing Detection Using Deep Learning. Future Generation Computer Systems, 109, 175-186.

Sahafizadeh, E., & Salari, S. (2020). Phishing Email Detection Using Natural Language Processing and Machine Learning. Information Sciences, 537, 387-403.

Sharifi, M., & Siadati, S. H. (2018). Dynamic Blacklist Approach for Phishing Website Detection. Journal of Cybersecurity Research, 7(3), 125-140.

Usuff, R., Manikandan, N., Kumaran, U. S., & Niveditha, C. (2017). Pattern Matching-Based Phishing Detection System Using Machine Learning Techniques. Cybersecurity and Applications, 6(1), 97-113.

Xie, J., Li, H., & Zhao, Y. (2020). Deep Learning-Based Phishing Website Detection Using Convolutional Neural Networks. IEEE Access, 8, 123435-123445.

Zhao, H., Zhang, K., & Liu, B. (2017). Random Forest-Based Phishing Detection for Web Security. Security and Communication Networks, 2017, 1-12.

APPENDIX

- **FEATURES CHECKING OF THE URL DATASET**

```
import pandas as pd

# Load dataset
df = pd.read_csv("cleaned_phishing_urls.csv")

# Check class distribution
print(df['class'].value_counts())  # 0 = Legitimate, 1 = Phishing

# View first few rows
print(df.head())
```

- **READING THE URL DATASET**

```
import pandas as pd

# Load CSV file from local path (update path accordingly)
df = pd.read_csv("phishing_url_website.csv")

# View first 5 rows
print(df.head())
```

- **EXTRACT THE FEATURES OF THE URL DATASET**

```
import pandas as pd
from urllib.parse import urlparse

# Load dataset
df = pd.read_csv("phishing_urls.csv")  # Replace with your actual dataset file

# Feature extraction function
def extract_features(url):
    parsed_url = urlparse(url)
    return {
        "url_length": len(url),
        "num_dots": url.count("."),
        "num_hyphens": url.count("-"),
        "num_at": url.count("@"),
        "https": 1 if parsed_url.scheme == "https" else 0
    }

# Apply feature extraction
df_features = df["url"].apply(lambda x: pd.Series(extract_features(x)))

# Combine extracted features with labels
df_final = pd.concat([df_features, df["label"]], axis=1)  # Assuming "label"
column exists
print(df_final.head())
```

- **EMAIL DATASET FEATURE EXTRACTION**

```
import re
```

```
# Load email dataset
emails = pd.read_csv("phishing_emails.csv")  # Replace with actual dataset

# Function to extract email features
def extract_email_features(email_text):
    num_links = len(re.findall(r'http[s]?://', email_text))  # Count links
    num_suspicious_words =
len(re.findall(r"(urgent|verify|bank|account|password|click)", email_text,
re.IGNORECASE))
    return {
        "num_links": num_links,
        "num_suspicious_words": num_suspicious_words
    }

# Apply feature extraction
emails_features = emails["email_text"].apply(lambda x:
pd.Series(extract_email_features(x)))

# Combine features with labels
emails_final = pd.concat([emails_features, emails["label"]], axis=1)
print(emails_final.head())
```

- **IMPORTING NECESSARY LIBRARIES AND SPLITTING THE DATA**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load preprocessed dataset (assuming df_final is already cleaned & has extracted
features)
X = df_final.drop(columns=["label"])  # Features
y = df_final["label"]  # Target (0 = Legitimate, 1 = Phishing)

# Split into train (80%) and test (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Standardize features (Logistic Regression performs better when features are scaled)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- **TRAINING THE MODEL USING LOGISTIC REGRESSION**

```
# Train Logistic Regression Model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate model
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

- **SAVING THE MODEL FOR DEPLOYMENT**

```
import pickle

# Save the trained model
with open("phishing_model.pkl", "wb") as file:
    pickle.dump(model, file)

# Save the scaler (important for consistent feature scaling)
with open("scaler.pkl", "wb") as file:
    pickle.dump(scaler, file)
```

- **SETTING UP THE FLASK API**

```
pip install flask pandas scikit-learn pickle-mixin



from flask import Flask, request, jsonify
import pickle
import numpy as np
```

```
# Load the trained model and scaler
model = pickle.load(open("phishing_model.pkl", "rb"))
scaler = pickle.load(open("scaler.pkl", "rb"))

# Initialize Flask app
app = Flask(__name__)

# Define feature extraction function
def extract_features(url):
    """Extracts necessary features from a given URL for prediction."""
    from urllib.parse import urlparse

    parsed_url = urlparse(url)
    return [
        len(url),                  # URL length
        url.count("."),              # Number of dots in URL
        url.count("-"),              # Number of hyphens
        url.count("@"),                # Presence of '@' symbol
        1 if parsed_url.scheme == "https" else 0  # Is HTTPS?
    ]

# Define API route for URL phishing detection
@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.get_json()
        url = data.get("url", "")

        # Extract features
        features = np.array(extract_features(url)).reshape(1, -1)

        # Scale features using the same scaler from training
        features = scaler.transform(features)

        # Make prediction
        prediction = model.predict(features)[0]
```

```
        # Convert to readable output
        result = "Phishing" if prediction == 1 else "Legitimate"

        return jsonify({"url": url, "prediction": result})

    except Exception as e:
        return jsonify({"error": str(e)})

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

- **RUNNING THE FLASK API**

  python app.py

- **TEST THE API**

```
#CREATE A TEST.PY FILE
import requests

url = "http://127.0.0.1:5000/predict"
data = {"url": "http://example.com/login"}

response = requests.post(url, json=data)
print(response.json())  # Output: {'url': 'http://example.com/login', 'prediction': 'Legitimate'}
```

- **DEPLOYING ON GITHUB**

  A. Set Up a GitHub Repository

  Go to GitHub and create a new repository.

  Upload the following files:

  - app.py (Flask API)
  - phishing_model.pkl (Trained model)
  - scaler.pkl (Trained scaler)
  - requirements.txt (Create this file)

B. Create a requirements.txt File

This file lists the Python libraries needed for the project.
Create requirements.txt and add:

Flask

scikit-learn

pandas

numpy

pickle-mixin

gunicorn

- **DEPLOYING ON RENDER**

Go to Render and create an account.

Click New Web Service and connect your GitHub repo.

Select:

Build Command: pip install -r requirements.txt

Start Command: gunicorn app:app

Python Version: Use 3.9 or higher

Click Deploy and wait a few minutes.

Once done, Render will give you a URL like:

https://your-app-name.onrender.com/predict

- **BUILDING THE CHROME EXTENSION FOR THE PHISHING URL**

A user clicks the extension icon.

The extension grabs the current webpage URL.

It sends the URL to our Flask API for phishing detection.

The extension shows an alert if the site is phishing or safe.

#create a new folder - phishing_extension

phishing_extension/

│── manifest.json # Extension settings

│── popup.html # Simple UI

│── popup.js # Handles API request

│── style.css # Styles

│── icon.png # Extension icon (optional)

- **CREATING A MANIFEST.JSON FILE(CHROME EXTENSION SETTING)**

#Create a file **manifest.json** inside phishing_extension/ and add:

```
{
 "manifest_version": 3,
 "name": "Phishing Detector",
 "version": "1.0",
 "description": "Checks if a website is phishing or safe",
 "permissions": ["activeTab", "scripting"],
 "host_permissions": ["<all_urls>"],
 "action": {
  "default_popup": "popup.html",
  "default_icon": {
   "16": "icon.png",
```

```
    "48": "icon.png",
    "128": "icon.png"
  }
 }
}
```

- **CREATE A POPUP.HTML FOR THE CHROME EXTENSION**

  #Create a file **popup.html** inside phishing_extension/ and add:

```
<!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>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h2>Phishing Detector</h2>
  <p id="status">Click the button to check this website.</p>
  <button id="checkButton">Check Website</button>
  <script src="popup.js"></script>
</body>
```

```
</html>

#Adding a style.css file
body {
    font-family: Arial, sans-serif;
    text-align: center;
    width: 200px;
    padding: 10px;
}

button {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px;
    margin-top: 10px;
    cursor: pointer;
    border-radius: 5px;
}

button:hover {
    background-color: #0056b3;
```

```
}
```

- **CREATE A POPUP.JS FILE FOR HANDLING API REQUESTS**

  Create a file **popup.js** inside phishing_extension/ and add:

```
document.getElementById("checkButton").addEventListener("click", () => {
    chrome.tabs.query({ active: true, currentWindow: true }, (tabs) => {
        let url = tabs[0].url;

        fetch("https://your-api-url.onrender.com/predict", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({ url: url })
        })
        .then(response => response.json())
        .then(data => {
            document.getElementById("status").innerText = `This site is: ${data.prediction}`;
        })
        .catch(error => {
            document.getElementById("status").innerText = "Error checking the site.";
        });
```

```
    });
});
```

Replace "https://your-api-url.onrender.com/predict" with your actual API URL.

- **LOAD AND TEST THE CHROME EXTENSION**

  Open Google Chrome.

  Go to chrome://extensions/.

  Turn on Developer Mode (top-right corner).

  Click "Load unpacked", then select the phishing_extension/ folder.

  The extension should now appear in the Chrome toolbar.

  Open any website and click the extension to check if it's phishing or safe!

- **PUBLISHING THE CHROME WEB EXTENSION**

  Create a Developer Account on Chrome Web Store

  Go to the Chrome Web Store Developer Dashboard

  Click Sign Up

  Accept the terms and conditions

  #Uploading the chrome web extension

  Click "New Item"

  Upload the ZIP file containing your extension

  Fill in the required details:

  Title: "Phishing Detector - Secure Browsing"

  Description: Explain how your extension helps detect phishing sites

Category: Choose Security & Privacy

Screenshots: Add at least one screenshot (recommended size: 1280x800 px)

Privacy Policy: If needed, you can create a simple privacy policy and host it on GitHub Pages or a free website

#Submit for Review

Click "Submit for Review"

Google will review your extension (this may take a few days)

If approved, your extension will be live on the Chrome Web Store!

- **BUILDING THE PHISHING EMAIL DETECTION**

  #EXTRACT KEY FEATURES FROM EMAIL

  #Modify preprocess.py to extract email features

```
import re
import pandas as pd
from email import message_from_string
from sklearn.feature_extraction.text import TfidfVectorizer


# Load Email Dataset
emails = pd.read_csv("emails.csv")


# Function to Extract Email Features
def extract_email_features(email_text):
    msg = message_from_string(email_text)
```

```
    # Extract Subject Line
    subject = msg["Subject"] if msg["Subject"] else ""

    # Extract Links in Email
    links = re.findall(r"https?://[^\s]+", email_text)

    # Check if Email Contains "Urgent", "Verify", etc.
    phishing_words = ["urgent", "verify", "click here", "account suspended", "reset password"]
    phishing_score = sum(word in email_text.lower() for word in phishing_words)

    return subject, len(links), phishing_score

# Apply Feature Extraction
emails["subject"], emails["num_links"], emails["phishing_score"] = zip(*emails["text"].apply(extract_email_features))

# Convert Email Body Text to TF-IDF Features
vectorizer = TfidfVectorizer(max_features=500)
email_features = vectorizer.fit_transform(emails["text"])
```

- **TRAINING THE EMAIL PHISHING DETECTION USING LOGISTIC REGRESSION**

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score


# Split Data into Training & Testing

X_train, X_test, y_train, y_test = train_test_split(email_features, emails["label"], test_size=0.2, random_state=42)


# Train Logistic Regression Model

model = LogisticRegression()

model.fit(X_train, y_train)


# Evaluate Model Accuracy

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.2f}")
```

- **Create an Email Scanner API**

```
#Modify app.py (Flask API for Email Detection)

from flask import Flask, request, jsonify

import joblib


app = Flask(__name__)


# Load the Trained Model
```

```
email_model = joblib.load("email_phishing_model.pkl")

@app.route("/predict_email", methods=["POST"])
def predict_email():
    data = request.get_json()
    email_text = data.get("email_text", "")

    # Extract Features
    subject, num_links, phishing_score = extract_email_features(email_text)
    email_features = vectorizer.transform([email_text])

    # Predict Phishing
    prediction = email_model.predict(email_features)[0]

    return jsonify({"prediction": "Phishing" if prediction == 1 else "Safe"})

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

- **INTEGRATE EMAIL SCANNER WITH GMAIL API**

  To scan incoming Gmail emails, we need Google's Gmail API.

Steps to Enable Gmail API

Go to Google Cloud Console

Enable Gmail API

Download OAuth Credentials (credentials.json)

Install required libraries:

pip install google-auth google-auth-oauthlib google-auth-httplib2 google-api-python-client

- **READ EMAIL AND DETECT PHISHING**

Modify gmail_scanner.py to fetch Gmail emails and classify them:

```
from googleapiclient.discovery import build

from google_auth_oauthlib.flow import InstalledAppFlow

import base64

import email


SCOPES = ["https://www.googleapis.com/auth/gmail.readonly"]


# Authenticate & Connect to Gmail

def gmail_authenticate():

    flow = InstalledAppFlow.from_client_secrets_file("credentials.json", SCOPES)

    creds = flow.run_local_server(port=0)

    return build("gmail", "v1", credentials=creds)
```

```
# Fetch Emails & Classify as Phishing or Safe

def scan_inbox():

    service = gmail_authenticate()

    results = service.users().messages().list(userId="me", labelIds=["INBOX"],
maxResults=5).execute()

    messages = results.get("messages", [])



    for msg in messages:

        msg_data = service.users().messages().get(userId="me",
id=msg["id"]).execute()

        msg_payload = msg_data["payload"]["body"]["data"]



        email_text = base64.urlsafe_b64decode(msg_payload).decode("utf-8")



        # Call the API to Detect Phishing

        response = requests.post("http://localhost:5000/predict_email",
json={"email_text": email_text})

        prediction = response.json().get("prediction")



        print(f"✉ Email Classified as: {prediction}")



scan_inbox()
```

- **Deploy the Email Detection API on a Cloud Server**

  Deploy on Render

  Create a free account at Render.com

  Upload your Flask project to GitHub

  Go to Render, click "New Web Service", and connect your GitHub repo

  Set the Start Command:

  gunicorn app:app

  Deploy! Your API will be live at:

  https://your-app-name.onrender.com/predict_email

  Deploy on AWS or Google Cloud

  Use EC2 (AWS) or Google Cloud Run

  Install Flask & run the app

  Set up a public URL with NGINX or Gunicorn

- **Create a Chrome Extension for Email Phishing Detection**

  FOLDER STRUCTURE

  email_phishing_extension

  ├── manifest.json

  ├── popup.html

  ├── popup.js

  ├── background.js

├── style.css

├── icon.png

#DEFINE MANIFEST.JSON

```
{
 "manifest_version": 3,
 "name": "Email Phishing Detector",
 "version": "1.0",
 "description": "Detects phishing emails in Gmail",
 "permissions": ["storage", "activeTab", "scripting"],
 "host_permissions": ["https://mail.google.com/*"],
 "background": {
  "service_worker": "background.js"
 },
 "action": {
  "default_popup": "popup.html"
 },
 "icons": {
  "128": "icon.png"
 }
}
```

#CREATE POPUP.HTML FOR USER INTERFACE

```
<!DOCTYPE html>
<html>
<head>
  <title>Phishing Detector</title>
  <script src="popup.js"></script>
  <style>
    body { width: 300px; padding: 10px; font-family: Arial, sans-serif; }
    button { padding: 10px; width: 100%; cursor: pointer; }
  </style>
</head>
<body>
  <h3>Check for Phishing</h3>
  <button id="scan">Scan Email</button>
  <p id="result"></p>
</body>
</html>
```

```
#CREATE A POPUP.JS (DETECTS EMAIL PHISHING)
document.getElementById("scan").addEventListener("click", function() {
    chrome.tabs.query({ active: true, currentWindow: true }, function(tabs) {
        let tab = tabs[0];
        chrome.scripting.executeScript({
```

```
            target: { tabId: tab.id },
            function: extractEmailText
        });
    });
});

function extractEmailText() {
    let emailBody = document.querySelector("div[role='main']").innerText;

    fetch("https://your-api-url.onrender.com/predict_email", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ email_text: emailBody })
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("result").innerText = `Result: ${data.prediction}`;
    });
}

#Create background.js (Manages Extension Behavior)
chrome.runtime.onInstalled.addListener(() => {
```

```
    console.log("Phishing Detector Installed!");
});


#Install & Test the Chrome Extension
#Open Chrome and go to chrome://extensions/
#Enable Developer Mode (top right corner)
#Click "Load Unpacked" and select the email_phishing_extension folder
#Open Gmail, click the extension, and scan an email.
#UPLOAD YOUR CHROME EXTENSION AND REVIEW IT
```

- **PERFORMANCE EVALUATION**

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Make predictions
y_pred = model.predict(X_test)

# Compute performance metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)


# Display results

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1 Score: {f1:.4f}')


# Confusion Matrix

cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Legitimate',
'Phishing'], yticklabels=['Legitimate', 'Phishing'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()
```