DEVELOP AN AI SYSTEM TO IDENTIFY AND BLOCK PHISHING EMAILS

A PROJECT REPORT

Submitted by

ANGESHKUMAR P 71052202301

ANISUTHAN S 71052202007

NATHESH P V 71052202303

SANJAYRAM S 71052202208

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COIMBATORE INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

COIMBATORE – 641 109

MAY 2025

COIMBATORE INSTITUTE OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certificate that this project report "**Experimental Analysis of Performance and Emission Characteristics of a Diesel Engine By Using Diesel -Tung Oil Blends**" is the Bonafide work of

ANGESHKUMAR P	**71052202301**
ANISUTHAN S	**71052202007**
NATHESH P V	**71052202303**
SANJAYRAM S	**71052202208**

who carried out the project work under my supervision

SIGNATURE	**SIGNATURE**
Dr .K. PUSHPALATHA, M.E,PH.D.,	**Ms .V. SARANYA,M.E**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	**ASSISTANT PROFESSOR**
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Coimbatore Institute of Engineering and Technology, Coimbatore 641 109	Coimbatore Institute of Engineering and Technology, Coimbatore 641 109

Submitted for the end semester project viva voce held on _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all, we extend our heartfelt gratitude to **Dr. K. A. CHINNARAJU**, **MBA., Ph.D.,** Director and the management of Coimbatore Institute of Engineering and Technology, for providing us with all sorts of support in the completion of this project. We, have great and immense pleasure in expressing the acknowledgment to the numerous contributors for the success of this project work.

We record our intentness to **Dr. K. MANIKANDA SUBRAMANIAN, M.E., Ph.D.,** Principal, Coimbatore Institute of Engineering and Technology for his guidance and encouragement for the successful completion of this Project.

We express my sincere thanks to **Dr. K. PUSHPALATHA, M.E., PH.D.,** Professor and Head, Department of Computer Science and Engineering for her constant encouragement and support throughout the course, especially for the useful suggestions given during the course of the project period.

We are highly grateful to **Ms. V. SARANYA**,**M.E.,**Assistant Professor, Department of Computer Science and Engineering, for her valuable suggestions and guidance throughout the course of this project. Her positive approach had offered incessant help in all possible ways from the beginning.

We also extend our sincere thanks to all the faculty members and non-teaching staffs of Department of Computer Science and Engineering and who have rendered their valuable help in completing this Project successful.

ABSTRACT

Phishing attacks are a significant cybersecurity concern that exploit human vulnerability by impersonating legitimate entities to steal sensitive information. These emails often contain malicious links or attachments that, when engaged with, compromise personal and organizational data. As cybercriminals use increasingly sophisticated techniques to craft deceptive messages, traditional filtering methods have become less effective. This project focuses on developing a machine learning-based system that can intelligently detect phishing emails to enhance email security and protect users from digital fraud.

The project uses a supervised learning approach to classify emails as phishing or legitimate (ham) by Analysing their textual content. A labelled dataset of emails is Pre-processed using Natural Language Processing (NLP) techniques, including tokenization, stop-word removal, and feature extraction using the TF-IDF method. Various machine learning models—such as Logistic Regression, Naïve Bayes, and Random Forest—are trained and evaluated on this processed data. Performance is measured using metrics like accuracy, precision, recall, and F1-score, with some models achieving over 95% accuracy, demonstrating the effectiveness of machine learning in phishing detection.

A simple web interface built using Flask enables users to test the system in real-time by inputting email text and receiving instant predictions. This practical implementation illustrates how the system can be integrated into email platforms for proactive phishing prevention. The study concludes that machine learning offers a scalable, adaptive, and accurate solution to phishing detection. Future work may explore deeper neural network models, real-time monitoring, support for multiple languages, and broader integration with cybersecurity infrastructure to further improve protection against evolving phishing threats.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ACRONYM	ABBREVIATION
1	AI	Artificial Intelligence
2	ML	Machine Learning
3	NLP	Natural Language Processing
4	CSV	Comma-Separated Values
5	HTML	Hyper Text Markup Language
6	CSS	Cascading Style Sheets
7	URL	Uniform Resource Locator
8	IP	Internet Protocol
9	TF-IDF	Term Frequency-Inverse Document Frequency
10	SVM	Support Vector Machine
11	LR	Logistic Regression
12	NB	Naïve Bayes
13	ROC	Receiver Operating Characteristic
14	API	Application Programming Interface
15	GUI	Graphical User Interface
16	HTTP	Hyper Text Transfer Protocol
17	JS	JavaScript
18	UI	User Interface
19	FN	False Negative
20	FP	False Positive
21	TP	True Positive
22	TN	True Negative
23	CSV	Comma-Separated Values
24	IDE	Integrated Development Environment
25	DNS	Domain Name System
26	SQL	Structured Query Language

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the rapidly expanding digital world, email remains one of the most widely used forms of communication, both in personal and professional domains. Unfortunately, this widespread usage has also made it a primary target for cybercriminals. Phishing, a type of cyberattack where attackers impersonate legitimate organizations to steal sensitive information, has emerged as a major threat to data privacy and cybersecurity. Phishing emails often appear authentic and can deceive even cautious users, resulting in severe financial losses and identity theft.

Traditional rule-based email filters are increasingly ineffective due to the evolving complexity of phishing tactics. To address this challenge, the use of Machine Learning (ML) offers a dynamic and scalable solution. By analysing large datasets of email content and behaviour patterns, ML models can learn to identify subtle cues and anomalies that indicate phishing attempts. Natural Language Processing (NLP) techniques further enhance this process by enabling systems to understand and analyse the semantics of email text.

This project aims to develop a machine learning-based phishing detection system capable of automatically classifying emails as legitimate or phishing. The solution includes collecting and preprocessing email datasets, extracting relevant features, training multiple ML algorithms, and evaluating their performance. Finally, the most accurate model is deployed through a simple web application to demonstrate real-time phishing detection.

The system not only helps in improving email security but also contributes to the broader goal of building intelligent cybersecurity tools that adapt to emerging threats. Through this project, we aim to showcase the practical implementation of machine learning in combating one of today's most prevalent cybercrimes.

1.2 PROBLEM STATEMENT

Phishing attacks continue to be one of the most dangerous and widespread cyber threats, with the potential to cause severe financial and reputational damage to individuals and organizations alike. Traditional email filtering systems, which often depend on predefined keyword lists, regular expressions, and static rule-based approaches, are largely ineffective in detecting modern phishing tactics. These systems may fail to recognize subtle and sophisticated phishing attempts that employ novel linguistic techniques, complex social engineering methods, and the use of images or hidden links to disguise their intent. As phishing methods become more advanced, these outdated systems struggle to adapt and adequately protect users.

One of the main limitations of conventional email filtering systems is their reliance on a fixed set of criteria for identifying phishing attempts. These criteria often involve searching for specific keywords or phrases commonly associated with phishing emails, such as "urgent," "click here," or "account verification." However, attackers continuously modify their strategies to evade detection by altering the text, structure, and appearance of their emails. This includes techniques such as email spoofing (disguising the sender’s address to appear legitimate), URL obfuscation (using misleading or disguised links), and domain impersonation (creating fake websites that look nearly identical to real ones).

Another key issue contributing to the success of phishing campaigns is the lack of user awareness. Many individuals still struggle to recognize phishing attempts, especially as attackers increasingly mimic trusted institutions and use realistic-looking designs. This lack of awareness, combined with the growing sophistication of phishing tactics, leads to a high success rate for cybercriminals. As a result, users continue to fall victim to phishing attacks, resulting in data breaches, identity theft, and financial loss.

To address these challenges, this project proposes the development of an intelligent, machine learning-based email filtering system capable of accurately distinguishing phishing emails from legitimate ones. By leveraging various machine learning algorithms and utilizing both text-based and structural features from email content, the system will offer a more robust approach to phishing detection. Key features to be analysed include the email's subject line, body content, presence of malicious URLs, attachments, and the structure of the email, all of which are important indicators of phishing attempts. The goal is to build a model that continuously evolves, learns from new patterns of phishing attempts, and provides near-instant detection with minimal user input.

The primary challenge lies in training a model that can generalize well to various phishing scenarios while minimizing false positives (legitimate emails mistakenly marked as phishing). This requires careful data collection, preprocessing, and feature engineering to ensure that the machine learning model captures the key characteristics of phishing emails, such as suspicious language patterns, unusual metadata, and hidden links. Additionally, the solution needs to be scalable, efficient, and easy to integrate with existing email systems, ensuring a seamless user experience without compromising performance or security.

In summary, the problem addressed in this project is the inadequacy of traditional, rule-based email filtering systems in detecting evolving phishing tactics. By applying machine learning techniques, the goal is to create a dynamic and adaptive phishing detection system that can analyse both the content and structure of emails to accurately classify them as either legitimate or phishing, ultimately reducing the risk of falling victim to such attacks.

1.3 OBJECTIVE

1.3.1 GENERAL OBJECTIVE

The primary objective of this project is to design and develop an intelligent phishing email detection system that leverages machine learning algorithms to automatically classify emails as either phishing or legitimate. With the increasing sophistication of phishing techniques that easily bypass traditional rule-based filters, there is a critical need for adaptive systems that can learn from data and evolve with new threats. This project focuses on applying natural language processing (NLP) and supervised learning methods to extract meaningful patterns from email text and metadata.

The system aims to provide real-time analysis of incoming emails by identifying deceptive content, suspicious links, and anomalies in email headers or formatting. By training the model on a diverse dataset of phishing and non-phishing emails, it is expected to generalize well across various email structures and phishing techniques. The goal is to significantly reduce the false positives and false negatives encountered in traditional spam filters and provide an additional security layer for individuals and organizations.

Moreover, this intelligent detection system is intended to be scalable, lightweight, and easy to integrate into existing email platforms. It not only contributes to minimizing financial and data loss due to phishing but also educates users by flagging and explaining potentially harmful emails. The project aims to support the broader goal of cybersecurity by enhancing trust and safety in digital communication through automated and accurate phishing detection.

1.3.2 SPECIFIC OBJECTIVES

1. To collect and preprocess a comprehensive dataset

Gather a balanced dataset containing a sufficient number of both phishing and legitimate (ham) emails from reliable sources such as public repositories and simulated environments. The data will be cleaned and pre-processed to remove HTML tags, special characters, and irrelevant metadata to improve model performance.

2. To apply Natural Language Processing (NLP) techniques

Utilize NLP preprocessing steps including tokenization, lemmatization, stop word removal, and feature extraction techniques like Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). This transforms raw email text into numerical representations that can be used effectively by machine learning algorithms.

3. To implement and compare multiple machine learning algorithms

Train various supervised machine learning classifiers such as Logistic Regression, Naïve Bayes, Support Vector Machine (SVM), and Random Forest. Each algorithm will be trained and tuned to identify which yields the best results for phishing detection based on the dataset characteristics.

4. To evaluate the performance of each model

Measure model effectiveness using standard performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. This comparison ensures the selected model is not only accurate but also reliable in distinguishing phishing emails from legitimate ones.

5. To design a user-friendly interface

Develop a frontend interface, preferably as a web or desktop application using technologies such as Flask (for backend) and HTML/CSS (for frontend), to allow users to input email content manually or through file upload. The system should return immediate feedback indicating whether the email is phishing or safe.

6. To reduce false positives and false negatives

Fine-tune hyperparameters and employ cross-validation to ensure the model minimizes misclassification. Reducing false positives ensures legitimate emails aren't wrongly flagged, while reducing false negatives ensures phishing emails aren't missed.

7. To assess the system's robustness

Evaluate how well the system performs on new or unseen email samples by introducing variations or obfuscated phishing techniques. This testing will demonstrate how well the model generalizes beyond its training data and adapts to evolving phishing patterns.

8. To provide recommendations for integrating the model into existing infrastructures

Suggest deployment strategies for real-time phishing detection, such as embedding the model in email servers or enterprise security tools. Recommendations will also include maintenance plans, update mechanisms, and potential integration with threat intelligence platforms.

1.4 CHAPTER WISE SUMMARY

1.4.1 INTRODUCTION

This chapter lays the foundation for the project by highlighting the increasing reliance on email communication and the corresponding rise in phishing attacks, which aim to steal sensitive information through deceptive messages. It explains why traditional spam filters and rule-based systems are insufficient to cope with evolving phishing strategies. The chapter presents the problem statement, emphasizing the need for an intelligent detection mechanism. It also sets forth the general and specific objectives of the study, such as improving detection accuracy and reducing false positives. The scope is defined to focus primarily on text-based analysis of emails. A brief overview of the machine learning approach, including model training, evaluation, and deployment, is provided. The chapter concludes with an outline of how the rest of the report is organized.

1.4.2 LITERATURE SURVEY

This chapter provides a comprehensive review of existing work in the domain of phishing detection. It begins with a detailed explanation of phishing techniques, including deceptive phishing, spear phishing, clone phishing, and whaling. The limitations of traditional methods such as blacklisting, whitelisting, and heuristic filters are discussed. The chapter then transitions to the role of machine learning in cybersecurity, covering popular models like Logistic Regression,

Naïve Bayes, SVM, Decision Trees, and Random Forest. Key past studies are analyzed to understand the strengths and weaknesses of these models. The comparative analysis section presents a side-by-side performance evaluation of various algorithms based on accuracy, precision, recall, and F1-score, establishing a rationale for selecting certain models in the current project.

1.4.3 METHODOLOGY

This chapter details the systematic approach used to build the phishing detection system. It begins with data acquisition, describing sources like public phishing email datasets and legitimate email samples. The preprocessing steps include data cleaning, text normalization, tokenization, removing stop words, and stemming or lemmatization. Feature extraction is performed using TF-IDF vectorization, transforming textual content into numerical features. Several machine learning algorithms are implemented and trained using a portion of the data. The models are validated using train-test splits and k-fold cross-validation. Performance is assessed using metrics such as accuracy, precision, recall, F1-score, and confusion matrix analysis. The methodology ensures the reproducibility and scalability of the detection system.

1.4.4 SYSTEM DESIGN AND IMPLEMENTATION

This chapter focuses on the system architecture and the practical implementation of the proposed solution. It describes the layered structure consisting of data input, model inference, and user interface components. The backend involves model loading, preprocessing scripts, and the prediction engine, all built using Python and machine learning libraries like Scikit-learn. The frontend is developed using Flask for web-based deployment, with simple forms where users can paste email content. The classification result is displayed as either "Phishing" or "Legitimate," along with confidence scores. The chapter also addresses error handling, model update mechanisms, and potential integration points for deployment into enterprise systems or email clients.

1.4.5 RESULTS AND ANALYSIS

This chapter presents the experimental results and evaluates the effectiveness of the proposed models. Each algorithm’s performance is shown using tables, graphs, and ROC curves. The best-performing model (e.g., Random Forest or SVM) is highlighted, and its precision-recall balance is discussed. A confusion matrix is used to identify areas of improvement, such as reducing false positives. Real-world test cases are analysed to showcase the system’s robustness and ability to generalize. The chapter also discusses execution time, model complexity, and scalability, offering

insights into the practicality of deploying the system on larger datasets or in real-time environments.

1.4.6 CONCLUSION AND FUTURE WORK

This chapter summarizes the achievements of the project, confirming that machine learning offers a reliable and scalable solution to phishing detection. It reaffirms that the selected algorithms successfully classified emails with high accuracy and low false detection rates. The model's adaptability to evolving threats makes it a valuable addition to cybersecurity toolkits. In terms of future work, the chapter proposes the use of deep learning techniques like LSTM and BERT for improved semantic understanding, integration of image and metadata analysis to catch more sophisticated phishing attempts, and development of browser-based or email client plug-ins for real-time usage. Additional research into adversarial attacks and multilingual phishing detection is also recommended.

CHAPTER 2

LITERATURE SURVEY

The increasing sophistication of phishing attacks has led to an urgent need for more effective methods of detection. Traditional techniques like rule-based filtering and blacklisting have proven insufficient in the face of evolving phishing tactics. In recent years, machine learning (ML) has emerged as a promising solution to enhance phishing detection systems by learning complex patterns from large datasets. This literature survey reviews various techniques and methods that have been proposed and applied in the domain of phishing email detection using machine learning.

2.1 FEATURE ENGINEERING

Feature engineering is a critical process in the development of machine learning based phishing detection systems, as it directly affects the performance and reliability of the predictive models. The primary goal is to extract, transform, and select relevant features from raw email data to allow the algorithms to differentiate between phishing and legitimate messages effectively. This step bridges the gap between unstructured email data and structured input required by most machine learning classifiers.

Researchers have identified and employed several categories of features for phishing detection, broadly classified into **lexical**, **URL-based**, **content-based**, and **header-based** features:

- **Lexical features** analyse the structure of the email content. Phishing emails often contain excessive use of urgent-sounding words (e.g., “urgent”, “verify”, “act now”), suspicious phrases (e.g., “reset your password”, “unauthorized login attempt”), and capitalization to trigger emotional responses. These features may also include the frequency of punctuation marks like “!”, “@”, or the use of long, obfuscated words and excessive whitespace.

- **URL-based features** focus on the characteristics of hyperlinks in emails. Attributes such as the presence of IP addresses in URLs instead of domain names, use of HTTPS or HTTP, length of URLs, and the number of subdomains are commonly analysed. Phishing URLs tend to be longer, often use URL shortening services (e.g., bit.ly, tinyurl), and may include slight spelling variations of legitimate domains (e.g., “go0gle.com” instead of “google.com”).

- **Header-based features** are extracted from email metadata and include sender-recipient address mismatch, anomalies in "Reply-To" fields, SPF (Sender Policy Framework) and

DKIM (DomainKeys Identified Mail) authentication results, and time of sending. These are useful in identifying spoofed emails or emails that originate from unusual IP addresses.

- **Content-based features** utilize the actual body text and subject line of the email. They are often the most informative for machine learning models. These features include the number of words, character density, entropy (randomness in text), HTML tags usage, presence of login forms, or embedded scripts.

With the rise of **Natural Language Processing (NLP)**, content-based feature engineering has seen significant advancements. Traditional approaches involve **Bag of Words (BOW)** and **TF-IDF (Term Frequency–Inverse Document Frequency)** for converting email content into numerical feature vectors. These methods capture word frequency and document relevance but lack semantic understanding.

To overcome these limitations, modern phishing detection systems have begun incorporating **contextual word embeddings** such as **Word2Vec**, **GloVe**, and more recently, **transformer-based embeddings** like **BERT (Bidirectional Encoder Representations from Transformers)** and **ELMo (Embeddings from Language Models)**. These methods understand word meanings in context, allowing the system to differentiate between benign and suspicious uses of the same word (e.g., "account" in a login notification vs. a marketing email).

In some cases, **character-level embeddings** are also used to detect obfuscation techniques where attackers substitute characters (e.g., "pa$$word" or "1ogin") to evade detection. Phishing campaigns also rely on slight grammatical errors, unnatural sentence constructions, or manipulative language, which can be captured by advanced NLP models trained to identify stylistic anomalies.

Furthermore, recent research suggests that **feature selection** techniques like **Chi-square**, **mutual information**, and **recursive feature elimination (RFE)** can improve model performance by retaining only the most predictive features. Dimensionality reduction methods such as **PCA (Principal Component Analysis)** are sometimes applied to reduce noise and improve model generalization.

Feature engineering in phishing detection has evolved from simple keyword detection to sophisticated NLP and statistical analysis. The effectiveness of a phishing detection system largely hinges on the quality and diversity of its feature set, which allows machine learning models to make accurate and explainable predictions, even in the presence of evolving phishing tactics.

2.2 DATA AUGMENTATION TECHNIQUES

In phishing email detection, one of the most persistent challenges is the **class imbalance** in available datasets. Legitimate (ham) emails are vastly more common than phishing emails, which creates a skewed class distribution. Machine learning models trained on such imbalanced data are prone to bias, often favoring the majority class and failing to correctly classify phishing emails. This significantly reduces the detection rate of actual phishing attempts—posing serious risks in real-world applications.

To address this issue, researchers have adopted a variety of **data augmentation techniques**, aimed at artificially increasing the number and diversity of phishing samples. These techniques not only help balance the dataset but also improve the model's ability to generalize to novel and sophisticated phishing strategies.

Synthetic Oversampling Methods

SMOTE (Synthetic Minority Over-sampling Technique) is one of the most widely used oversampling techniques in phishing detection. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the minority class, and generating synthetic samples along that line. **ADASYN (Adaptive Synthetic Sampling)** is a variation that focuses on generating more samples in regions where the minority class is sparsely represented, enhancing focus on hard-to-learn examples. Both techniques have been shown to significantly improve recall in phishing classification tasks.

2.2.1 TEXT-BASED NLP AUGMENTATION TECHNIQUES

In recent years, **Natural Language Processing (NLP)**-oriented data augmentation methods have become increasingly popular due to the textual nature of email content:

- **Back-translation** is a powerful technique in which an email is translated to a different language (e.g., English → French → English) to produce a semantically similar but syntactically different version of the text. This helps introduce variation in word choice and phrasing without altering the original intent of the email.
- **Easy Data Augmentation (EDA)** techniques—such as **synonym replacement**, **random insertion**, **random swap**, and **random deletion**—are lightweight and effective. These strategies introduce minor perturbations in the text that simulate the diversity found in phishing emails.

- **Contextual augmentation** using **transformer-based models** (e.g., BERT or GPT) allows the generation of new text samples by predicting and replacing words in context, resulting in syntactically and semantically realistic phishing email variants.

2.2.2 GAN-BASED PHISHING EMAIL SYNTHESIS

More advanced augmentation involves using **Generative Adversarial Networks (GANs)** for text generation. GANs consist of a generator and a discriminator competing against each other, where the generator tries to create fake (but realistic) emails, and the discriminator attempts to distinguish real emails from synthetic ones. Studies such as Text-GAN and Seq-GAN have explored phishing email synthesis by training on real email datasets. These models can generate phishing emails that mimic human writing patterns, enabling the classifier to learn from a more diverse set of attack examples.

GAN-generated phishing emails are especially valuable for simulating **zero-day attacks**—new phishing tactics that have not yet been observed in the wild. By training classifiers on these simulated threats, researchers aim to build **proactive defence systems** capable of identifying phishing attempts that deviate from known patterns.

2.2.3 BENEFITS AND LIMITATIONS

The primary benefit of data augmentation is the **enhanced robustness** and **generalization** capability of phishing detection models. Augmented data allows the classifier to better handle variation in writing styles, structures, and obfuscation tactics often used by cybercriminals.

However, there are limitations. Poorly designed augmentation can introduce noise or unrealistic patterns that may degrade performance. Textual data is sensitive to context, and careless alterations may change the meaning of the email content. Therefore, it is crucial to validate synthetic data through manual inspection or use human-in-the-loop systems to ensure quality.

Data augmentation is a powerful strategy for improving phishing detection systems, particularly when real-world phishing samples are limited. Techniques like SMOTE, ADASYN, NLP-based transformations, and GAN-based text generation contribute to creating a more balanced, diverse, and comprehensive training dataset. These advancements ultimately support the development of machine learning models that are not only accurate but also resilient against evolving cyber threats.

2.3 CLASSIFICATION ALGORITHMS

Machine learning (ML) has emerged as a powerful tool in combating phishing attacks, which are increasingly sophisticated and harder to detect using traditional rule-based filters. ML allows systems to automatically learn patterns from data, identify anomalies, and make decisions without being explicitly programmed for every possible phishing scenario.

In the context of phishing email detection, machine learning enables the classification of emails into two categories: **phishing** and **legitimate (ham)**. This is achieved by training models on labelled datasets containing both types of emails. The models learn from patterns in email content, metadata, and embedded links, helping to identify malicious emails with high accuracy.

By applying natural language processing (NLP) techniques, text content from emails is transformed into numerical features, allowing the ML models to understand linguistic patterns commonly used in phishing. Once trained, the models can generalize and classify unseen emails, making them effective tools in real-time email filtering systems.

Key Machine Learning Algorithms Used in the Project

Several ML algorithms were explored and evaluated for phishing detection. The key ones include:

In this project, a range of machine learning algorithms were evaluated to build an effective phishing email detection system. Logistic Regression was one of the first algorithms explored due to its simplicity and interpretability. As a binary classifier, it is useful in mapping word occurrences in emails to either the phishing or legitimate category. Similarly, the Naïve Bayes classifier, known for its effectiveness in spam detection, was considered because of its ability to perform well on text classification tasks by applying Bayes' Theorem and assuming feature independence, which often simplifies model training without sacrificing much accuracy.

The study also investigated Support Vector Machines (SVM), a powerful algorithm that identifies the optimal decision boundary (or hyperplane) to separate classes with the widest margin. This method works particularly well with high-dimensional input data such as the TF-IDF vectors derived from email content. In addition, Random Forest, an ensemble learning technique that combines the results of multiple decision trees, was tested for its robustness and ability to capture more complex, non-linear patterns within the data. Its ability to handle feature interactions and prevent overfitting makes it a strong candidate for phishing detection.

Furthermore, K-Nearest Neighbours (KNN) was included as a baseline algorithm. While it is conceptually simple and classifies inputs based on similarity to nearby training samples, its computational cost increases significantly with larger datasets. Lastly, the project utilized XG-Boost (Extreme Gradient Boosting), a state-of-the-art boosting algorithm that builds decision trees sequentially while correcting the errors of previous trees. Its speed, accuracy, and ability to handle large datasets efficiently made it one of the most promising models in the phishing detection pipeline. Overall, the combination of traditional and advanced algorithms allowed for comprehensive performance evaluation and informed the selection of the final model

How Models Identify Phishing Patterns by Analysing Characteristics

Machine learning models are trained on features extracted from emails to differentiate phishing from legitimate messages. These characteristics fall into several categories:

Phishing detection relies on a combination of content analysis, metadata inspection, and advanced statistical techniques. **Textual content features** play a key role in identifying malicious intent. Phishing emails often create a sense of urgency using words like “immediately,” “urgent,” or “verify now,” aiming to pressure recipients into acting without thinking. They frequently request sensitive information, such as passwords or account confirmations, and may contain noticeable spelling and grammar errors—less common in legitimate communications. Threatening language, such as warnings about account suspension, is also a typical sign of phishing.

URL features are another strong indicator. A mismatch between the visible link and its actual destination URL is a major red flag. Phishing emails often use suspicious or look-alike domains, such as "google.com" instead of "google.com," and may use insecure "http" links rather than the secure "https" protocol.

Email metadata features also provide valuable clues. The sender’s email address may appear spoofed or unfamiliar, and the "Reply-To" address often differs from the sender's, indicating a potential scam. Additionally, emails sent at odd hours like late at night or very early in the morning may suggest automation or malicious intent.

Statistical and structural features include characteristics such as email length and the number of embedded links or attachments. Phishing emails are typically short, generic, and heavily templated. The inclusion of multiple links or attachments can further increase suspicion. Advanced phishing attempts might use HTML content or even embedded forms and JavaScript to deceive recipients.

Finally, **NLP-based features** enhance detection through natural language processing. Techniques like Bag-of-Words and TF-IDF convert textual content into numerical vectors for analysis. Monitoring the frequency of specific phishing-related keywords and calculating sentiment scores can help uncover emotionally manipulative or persuasive language, which is commonly used to deceive users. Together, these features form a robust framework for identifying phishing attempts and protecting users from cyber threats.

These features are input into the ML models, which then classify the email as phishing or legitimate. During training, the models learn patterns such as the frequent occurrence of suspicious links, urgent language, and sender anomalies in phishing emails.

2.4 ENSEMBLE APPROACHES

Ensemble learning has emerged as one of the most effective strategies in phishing detection, offering improved generalization, robustness, and predictive performance by leveraging the combined strength of multiple learning algorithms. The core idea is that while individual models may make errors on different instances, their collective decision tends to be more accurate and stable.

Random Forest

Random Forest, one of the most widely used ensemble methods, builds numerous decision trees on bootstrapped samples and averages their predictions for classification. This technique effectively reduces variance and overfitting, which are common problems with single decision trees. In phishing detection, Random Forest benefits from its ability to handle high-dimensional, noisy datasets and deliver strong performance even with default hyperparameters. Its interpretability via feature importance scores also aids in understanding which features are most predictive.

Gradient Boosting and XG-Boost

Gradient Boosting techniques, including XG-Boost (Extreme Gradient Boosting), have set the benchmark for structured data classification problems. XG-Boost builds trees sequentially, where each new tree corrects the errors of the previous ones using gradient descent optimization. In phishing detection, it excels due to its regularization mechanisms, handling of missing data, and support for parallel processing. Studies have shown that XG-Boost consistently outperforms traditional methods when combined with engineered features from email content and metadata.

Light-GBM and Cat-Boost—newer gradient boosting frameworks—have also been explored in phishing research for their enhanced training speeds and ability to deal with categorical variables more effectively.

Voting Classifiers

Voting classifiers aggregate the predictions of a diverse set of base learners such as SVM, Logistic Regression, Naïve Bayes, and Decision Trees. Two primary voting strategies are used:

- **Hard Voting**: Each model votes for a class label, and the label with the majority votes is chosen.
- **Soft Voting**: Probabilities from each model are averaged or weighted before making the final prediction, often leading to better performance in imbalanced datasets.

Voting-based ensembles improve resilience against overfitting by combining strengths of linear, probabilistic, and tree-based models.

Stacking and Blending

More complex ensemble methods like **stacking** and **blending** are increasingly used in phishing detection. These methods involve training multiple base classifiers (level-0 models) and feeding their outputs into a meta-classifier (level-1 model) that learns how to best combine them. For instance, outputs from SVM, NB, and Random Forest might be used as input to a logistic regression meta-model. This layered architecture captures both low- and high-level patterns in the data.

Blending is a simplified version of stacking, usually involving a holdout validation set rather than cross-validation. While slightly less rigorous, it's easier to implement and often effective in practice.

Advantages in Phishing Detection

- **Improved Accuracy**: Ensemble methods reduce generalization errors by averaging out individual model weaknesses.
- **Better Handling of Complex Features**: By combining different types of learners, ensemble models can simultaneously capture linear, non-linear, and probabilistic patterns in email content and structure.
- **Robustness Against Noisy Data**: Phishing emails often contain deceptive language or irregular formatting. Ensemble models are better equipped to manage this variability.

- **Adaptability**: Ensembles can be retrained with updated base models to remain effective against evolving phishing strategies.

Limitations and Challenges

Despite their performance benefits, ensemble methods also introduce some challenges:

- **Increased Complexity**: More computational resources are needed for training and inference.
- **Reduced Interpretability**: It becomes harder to explain decisions made by stacked or boosted models compared to single models.
- **Deployment Overhead**: Real-time email filtering systems must balance accuracy with latency, which can be a constraint for heavier ensemble models.

Recent Trends

Recent literature shows that hybrid ensembles—such as combining deep learning models (e.g., LSTM) with boosting algorithms—offer promising results. Ensemble learning is also being integrated with transfer learning and adversarial training to create more resilient phishing detection systems that can adapt to unseen threats.

Applications of Ensemble Approaches in Phishing Detection

Ensemble methods are powerful strategies that combine the predictive strengths of multiple base models to improve accuracy, robustness, and generalization in phishing detection. Here's how they are applied:

1. Improved Classification Accuracy

- By aggregating the predictions of diverse models (e.g., Logistic Regression, SVM, Random Forest), ensemble methods reduce variance and bias.
- For example, Random Forest combines many decision trees, each trained on a random subset of features and data, to deliver more reliable predictions.

2. Handling Imbalanced Datasets

- Phishing datasets are often skewed toward legitimate emails. Ensemble techniques like **Bagging** (used in Random Forest) and **Boosting** (used in XGBoost or AdaBoost) can help better capture the minority class (phishing).

- Boosting methods emphasize harder-to-classify examples, which improves recall—a key concern in phishing detection.

3. Robustness to Noise and Variability

- Phishing emails vary in structure, language, and sophistication. Ensembles can generalize better across such variability, reducing overfitting and increasing detection of novel phishing patterns.

4. Hybrid Learning Systems

- Voting classifiers or stacked ensembles allow for combining linear models (e.g., Logistic Regression) with non-linear models (e.g., Random Forest) and even deep learning models (e.g., BERT), creating a hybrid system that balances interpretability and accuracy.

5. Context-Aware Predictions

- Ensemble models can integrate multiple feature types (content-based, header-based, URL-based) through parallel models and combine their outputs intelligently, improving context understanding.

Challenges of Ensemble Approaches in Phishing Detection

While ensemble models offer improved accuracy in phishing detection, they also introduce several challenges that must be carefully managed. One of the primary concerns is the **increased computational cost**. Training multiple models and aggregating their predictions demand significant CPU/GPU resources, along with substantial memory and storage. This becomes especially problematic in real-time detection scenarios where rapid decision-making is required, such as in large-scale enterprise email systems.

Model complexity is another major drawback. Ensemble methods often act as "black boxes," making it difficult to understand and explain their decisions. This lack of interpretability can hinder trust and usability, especially when security analysts or end users need clear reasons for why an email was flagged as phishing.

Additionally, ensemble approaches introduce significant **maintenance overhead**. These models require frequent updates to remain effective against evolving phishing tactics. Retraining and tuning multiple components within the ensemble is more complex and time-consuming compared to maintaining a single, standalone classifier.

There's also the **risk of overfitting**, particularly in stacking-based ensembles. If not properly validated, stacking or blending can lead to models that perform well on training data but fail to generalize to new, unseen samples—especially in cases involving small or noisy datasets common in phishing detection.

Finally, **deployment and integration complexity** presents real-world challenges. Implementing ensemble models in existing infrastructures, such as email filtering systems or security APIs, can be difficult due to additional dependencies, processing time, and system latency. Overall, while ensemble models offer performance benefits, they must be carefully balanced against these practical limitations.

Aspect	Details
Common Algorithms	Random Forest, XG-Boost, AdaBoost, Voting Classifiers, Stacking
Benefits	Improved accuracy, better handling of imbalance, robustness to variation
Applications	Hybrid classifiers, real-time threat detection, multi-feature integration
Challenges	High computation, interpretability, complex updates, risk of overfitting

Table 2.4.1: Ensemble Approaches in Phishing Detection

2.5 EXISTING METHODOLOGIES

Over the past two decades, phishing detection methodologies have evolved significantly from static rule-based systems to dynamic, data-driven AI approaches. As phishing tactics have become more sophisticated—employing tactics like email spoofing, domain obfuscation, and social engineering—traditional detection techniques have proven insufficient.

1. Traditional Approaches

Traditional phishing detection approaches relied heavily on methods such as **blacklist-based filtering**, **heuristic rules**, and **signature-based detection**. Blacklists work by blocking known malicious domains, IP addresses, or sender emails, while heuristic methods use handcrafted rules such as checking for the presence of "https," counting subdomains, or detecting suspicious keywords to flag potential threats. Signature-based detection, similarly, identifies phishing attempts by matching exact or partial patterns of previously known malicious content. Although these techniques are efficient and require minimal computational resources, they are increasingly ineffective against modern phishing attacks, as attackers frequently use **novel, polymorphic, or slightly altered messages** that easily bypass these static defences.

2. Machine Learning-Based Approaches

Modern systems use **supervised learning** to identify phishing emails by learning patterns from labelled data. Key features used include:

- Email body and subject content (TF-IDF, word embeddings)
- Header and metadata fields (e.g., sender address anomalies)
- Embedded URL features (e.g., URL length, IP use, redirect count)

Popular algorithms used include **Random Forest**, **SVM**, **Naïve Bayes**, and **XGBoost**. These methods are adaptable and can handle zero-day attacks better than rule-based systems, though they require regular model retraining and quality data preprocessing.

3. Deep Learning and Transformer Models

Deep learning approaches, especially **LSTM** and **CNN** architectures, have been employed for automatic feature extraction from text and URLs. These models capture temporal and spatial dependencies in email sequences and provide better generalization to unseen patterns.

More recently, **transformer-based models** like **BERT**, **RoBERTa**, and **DistilBERT** have outperformed conventional techniques in understanding nuanced and context-aware phishing language. These models are fine-tuned on phishing datasets and offer superior performance, especially when used with email-specific domain adaptation.

4. Hybrid and Multi-Layered Models

Hybrid and multi-layered models have become a state-of-the-art approach in phishing detection by combining various analytical techniques to enhance accuracy and robustness. These models integrate Natural Language Processing (NLP) for analyzing email content with URL inspection methods such as DNS checks and WHOIS data to assess the legitimacy of embedded links. Additionally, they leverage email metadata like sender reputation and timing alongside advanced natural language understanding to detect nuanced threats. Many frameworks also combine shallow models (e.g., Voting Classifiers) with deep learning architectures (e.g., RNNs or Transformers), achieving a balance between speed, interpretability, and detection power. This multi-layered strategy allows for more effective and adaptive real-time detection, making it especially suitable for large-scale enterprise email systems.

5.Metadata Analysis

Metadata analysis is a critical component in modern phishing detection systems, as it provides structural and behavioral cues about the origin, transmission, and content context of an email without solely relying on the email's textual content. Metadata includes information found in the email headers, such as the sender's IP address, "From" and "Reply-To" addresses, sending time, authentication results (e.g., SPF, DKIM, and DMARC status), and the mail client used. These features often reveal anomalies or inconsistencies that are strong indicators of phishing.

For example, **spoofed "From" addresses** that do not align with the legitimate sender domain or mismatch with the "Reply-To" address can suggest impersonation. Emails sent at **unusual hours** or from **geographically unlikely IPs** may indicate automation or bot-based phishing campaigns. Additionally, examining the **email path (Received headers)** can reveal irregularities in how the email was routed, especially if intermediate servers are not trusted or are located in high-risk regions.

Moreover, metadata fields can be quantitatively analyzed to build machine learning features such as frequency of occurrence, sender reputation scores, and historical legitimacy of the sending domain. Integrating this metadata with content-based and URL-based features in hybrid models significantly improves phishing detection accuracy. In enterprise systems, **metadata analysis also aids in forensic investigations**, helping trace the source and method of an attack, and contributing to automated threat intelligence sharing. Overall, metadata analysis is a lightweight, non-intrusive, yet powerful technique that enhances the precision and reliability of phishing detection systems.

These solutions offer the best of both worlds robust classification and real-time capability but may face **scalability** and **latency issues** in high-volume environments

Table 2.5.1: Comparison of Existing Phishing Detection Methodologies

Methodology	Description	Key Features	Advantages	Limitations	Use Case / Application
Blacklist-based	Checks URLs, IPs, or domains against known malicious lists	Uses predefined threat databases	Fast, simple, low computational cost	Ineffective for zero-day or obfuscated attacks	Email clients, browser extensions
Heuristic-based	Uses handcrafted rules (e.g., keyword matching, suspicious patterns)	Rule-driven, no training needed	Easy to implement, interpretable	Static, prone to evasion, high false positives	Spam filters, legacy email systems
Machine Learning	Trains models (SVM, Naïve Bayes, RF, etc.) on email content and metadata	Uses TF-IDF, BoW, handcrafted features	Adaptive, data-driven, scalable	Needs labeled data, sensitive to feature quality	Enterprise-grade email filtering systems
Deep Learning	Uses neural networks (CNNs, LSTMs) to extract features automatically	Sequence modeling, learns complex semantics	High accuracy, minimal feature engineering	Requires large datasets, high compute power	Cloud-based phishing detection engines
Transformer-based NLP	Uses pretrained language models (BERT, RoBERTa) fine-tuned for phishing emails	Contextual understanding, sentence embeddings	Best-in-class accuracy, understands email context	Resource intensive, slower inference	Modern NLP-enhanced enterprise platforms
Hybrid Models	Combines ML/DL with metadata, URL analysis, ensemble methods	Multi-layered: combines content, URL, header, and sender features	Robust, handles varied phishing strategies	Complex integration, higher development effort	Security suites, SOC tools
Ensemble Learning	Combines multiple classifiers (e.g., Voting, RF, XGBoost, Stacking)	Aggregates outputs from base models	Reduces variance, improves generalization	May increase latency, requires diverse model base	Threat intelligence platforms
GAN-based Augmentation	Uses Generative Adversarial Networks to synthesize phishing emails	Generates realistic phishing text for training	Improves generalization, helps handle class imbalance	Complex training process, risk of generating noisy samples	Model training pipelines, research experiments
URL Analysis Tools	Uses lexical + domain-based features to assess embedded URLs	Analyzes structure, domain age, HTTPS, redirects	Fast, effective against malicious links	May miss phishing with benign-looking URLs	Browser-integrated email protection
Metadata Analysis	Inspects email headers (e.g., SPF, DKIM, reply-to mismatch)	Checks sender legitimacy, routing, and authentication	Early-stage filtering, useful against spoofing	Can’t detect phishing from compromised legitimate accounts	Mail server-level screening and threat hunting

Key Challenges in Existing Systems

Phishing detection systems face the persistent challenge of **false positives**, where legitimate emails are incorrectly flagged as malicious. This not only disrupts important communications but also erodes user trust in the security system, potentially causing users to ignore or override warnings. Striking the right balance between catching phishing attempts and avoiding false alarms is essential to maintain both security and usability. Another major hurdle is the need for **real-time detection**. While deep learning models provide improved accuracy, their computational intensity can introduce delays, especially in large-scale email environments. To be effective, these models require optimization techniques such as pruning or using lightweight architectures to ensure timely identification and blocking of phishing emails before they reach users.

Additionally, many phishing detection models struggle with **generalization**, as they often overfit to specific datasets and fail to adapt well to diverse linguistic or domain-specific contexts. This is particularly problematic when dealing with phishing emails in multiple languages or specialized industries where terminology and attack methods vary widely. Addressing this requires diverse training data and adaptive learning strategies. Furthermore, **data privacy** concerns are paramount since phishing detection involves analyzing potentially sensitive email content. Compliance with regulations like GDPR and HIPAA necessitates privacy-preserving measures such as encryption, anonymization, or federated learning to protect user data while still effectively detecting threats. Balancing accuracy, speed, adaptability, and privacy remains a complex but vital challenge in the development of phishing detection systems.

CHAPTER 3

OPTIMIZATION ALGORITHMS

3.1 FLOWCHART AND DESCRIPTION

1.Start / Email Input

The phishing detection process begins when an email is received or selected from stored data. This email can come directly from a user's inbox or from an archive for batch analysis. It is submitted through a user interface or automatically via an email processing pipeline. The system prepares to analyze the email content and metadata. This ensures that every relevant message is scanned for phishing threats.

2.Data Preprocessing

The email content undergoes cleaning removing HTML tags, special characters, and converting text to lowercase. Then, Natural Language Processing (NLP) techniques such as tokenization, stop-word removal, stemming/lemmatization, and vectorization (e.g., TF-IDF) are applied.

3.Feature Extraction

Features are extracted from multiple parts of the email for thorough analysis. URL features such as the presence of IP addresses, domain length, and suspicious characters are identified. Text features include detecting phishing keywords, urgency phrases, and emotional triggers. Email header anomalies like mismatched "Reply-To" addresses or spoofed sender information are also examined. Additionally, embedded HTML and script elements, such as forms or JavaScript, are analyzed to detect hidden threats.

4.Model Selection / Classification

The extracted features are fed into machine learning models that have been trained to detect phishing patterns. Common classifiers include Logistic Regression, Naïve Bayes, Support Vector Machines (SVM), and Random Forests. More advanced systems use ensemble classifiers like voting or stacking to combine predictions from multiple models for greater accuracy. The model processes the input data and predicts whether the email is phishing or legitimate. This step is critical for identifying threats based on learned patterns.

5.Prediction Output

The model outputs a classification decision, labeling the email as either phishing or legitimate. For ensemble methods, the final decision is made through majority voting or weighted averaging of individual model predictions. This output informs the system on the appropriate next steps to take regarding the email. Accuracy at this stage is vital to reduce false positives and negatives. The prediction forms the basis for user alerts and security actions.

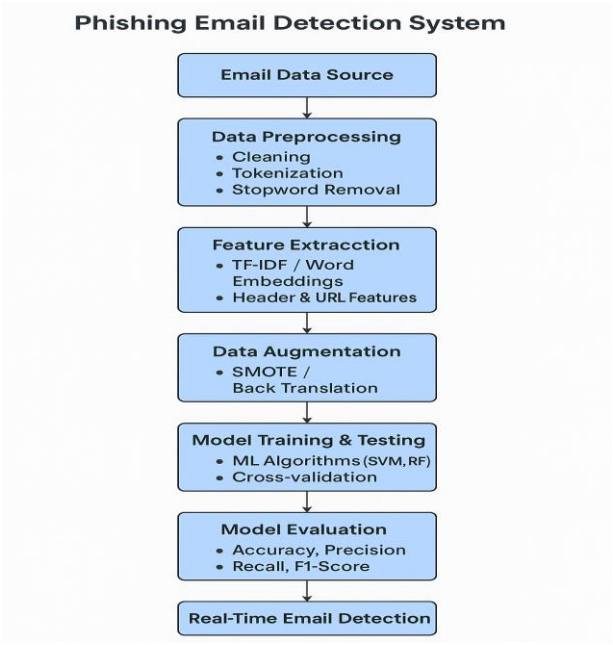
6.Display / Alert System

Based on the model's prediction, the system issues an alert if the email is suspected to be phishing. Users may receive warnings in their email client or through a security dashboard. If the email is safe, the user is notified or the message is delivered without interruption. The system logs the detection results for auditing and future model training. This step helps users stay informed and take necessary precautions against phishing.

7.End

The phishing detection cycle concludes with the results saved securely in the system database. These saved outcomes can be used periodically to retrain and improve the detection models. Continuous updates ensure the system adapts to new phishing techniques and evolving threats. The process then resets to analyze incoming emails in a similar manner. This completes one full loop of email phishing detection and defense.

Fig 3.1. Flowchart: Phishing Email Detection System



Description of Each Step

1. Email Data Source

The first step involves collecting a well-labelled dataset containing both phishing and legitimate (ham) emails. These datasets can be sourced from publicly available repositories like Kaggle, Phish-Tank, or through custom web scraping of email archives. Having a diverse and accurately labelled dataset is crucial for training models that can effectively differentiate phishing emails from legitimate ones. The dataset should represent various phishing tactics and legitimate email formats to ensure the model learns robust patterns.

2. Data Preprocessing

Before feeding emails into a model, the raw text data must be cleaned and standardized. This involves removing unwanted characters such as HTML tags, punctuation, and special symbols that do not contribute meaningfully to analysis. The entire email content is then converted to lowercase to ensure uniformity. Tokenization is applied to split the text into meaningful words or tokens, and stop-words common words with little semantic value like "and" or "the" are removed to reduce noise. These preprocessing steps help prepare the data for more effective feature extraction**.**

3. Feature Extraction

In this step, the cleaned text is transformed into numerical vectors using techniques like TF-IDF or advanced word embeddings such as Word2Vec or BERT. These representations capture the semantic meaning of the email content. Additionally, structural features are extracted from email headers, URLs, and metadata. Important indicators include the length of URLs, presence of IP addresses instead of domain names, number of special characters in links or text, and email authentication status such as SPF or DKIM records. Combining textual and structural features enables the detection system to identify subtle signs of phishing.

4. Data Augmentation

To address dataset imbalances where phishing samples are often fewer than legitimate emails, data augmentation techniques are applied. Synthetic sample generation methods like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN can create artificial phishing examples to balance the dataset. NLP-specific augmentation strategies such as synonym replacement or back-translation (translating text to another language and back) help increase data diversity without changing the underlying meaning. These techniques improve model robustness and reduce bias toward majority classes.

5. Model Training and Testing

Multiple machine learning classifiers are trained and tested to find the most effective model for phishing detection. Popular algorithms include Naïve Bayes, Logistic Regression, Random Forest, and Support Vector Machines (SVM). During training, k-fold cross-validation is used to split the dataset into multiple subsets, ensuring that the model's performance is evaluated on different portions of data for reliability. This step optimizes model parameters and prevents overfitting, leading to better generalization on unseen emails.

6. Model Evaluation

Once trained, the models are evaluated using metrics that reflect various aspects of performance. Accuracy measures the overall correctness of predictions, but can be misleading with imbalanced data. Precision indicates how many emails predicted as phishing are actually phishing, reflecting the system's ability to avoid false positives. Recall measures the proportion of actual phishing emails detected, highlighting the system's sensitivity. The F1-score combines precision and recall into a single metric, providing a balanced view of model effectiveness in identifying phishing emails.

7. Real-Time Email Detection

Once trained, the models are evaluated using metrics that reflect various aspects of performance. Accuracy measures the overall correctness of predictions, but can be misleading with imbalanced data. Precision indicates how many emails predicted as phishing are actually phishing, reflecting the system's ability to avoid false positives. Recall measures the proportion of actual phishing emails detected, highlighting the system's sensitivity. The F1-score combines precision and recall into a single metric, providing a balanced view of model effectiveness in identifying phishing emails.

3.2 DESIGN METHODOLOGY

Phishing email detection using machine learning involves a structured methodology that transforms raw email data into actionable predictions. The methodology is designed to ensure accuracy, adaptability to evolving threats, and efficient deployment in real-world scenarios such as enterprise email systems and personal email clients.

1. Data Collection

A robust dataset forms the backbone of an effective phishing detection system, as it ensures the model learns from diverse and representative examples. Publicly available datasets such as Spam Assassin, Phish-Tank, the Enron Email Dataset, and phishing-specific collections on Kaggle offer pre-labelled emails categorized as either legitimate or phishing. To further enhance dataset richness and variability, additional data can be sourced from online repositories, cybersecurity incident reports, and email honeypots traps set by security researchers to collect phishing attempts in the wild. During this data collection phase, various attributes are extracted to support comprehensive analysis. These include the email's subject and body content, sender addresses, domain names, reply-to headers, and embedded URLs. Other valuable features encompass email attachments, authentication headers like SPF, DKIM, and DMARC, as well as metadata such as timestamps, IP addresses, and geolocation (when available). This stage is essential for capturing the full spectrum of phishing strategies across different platforms, time periods, and threat actors, thereby improving the model's ability to generalize and detect novel phishing patterns.

2. Data Preprocessing

Data preprocessing is a critical phase in phishing detection as it transforms unstructured, raw email content into structured formats suitable for machine learning models. The process begins with text cleaning, where noisy elements such as ads, special characters, redundant whitespace, and scripts are removed to ensure textual consistency. HTML parsing follows, stripping away HTML tags, embedded JavaScript, and inline styles, allowing the system to focus solely on the meaningful textual content of the email. Tokenization breaks the cleaned content into individual words or tokens, and lowercasing helps reduce case sensitivity, which standardizes the data further. Normalization is performed through techniques like stemming reducing words to their root forms using algorithms like the Porter Stemmer and lemmatization, which is more context-sensitive and preserves grammatical accuracy, often implemented using libraries such as SpaCy or NLTK. Finally, vectorization techniques are applied to convert text into numerical features. TF-IDF highlights important words based on their frequency and uniqueness, Bag-of-Words (BoW) captures raw word occurrence counts, and word embeddings like Word2Vec and GloVe embed semantic relationships between words. More advanced models may use BERT embeddings to capture deeper contextual nuances in phishing language, enhancing detection performance.

3. Feature Engineering

Feature engineering plays a pivotal role in enhancing a phishing detection model's accuracy by enabling it to recognize patterns and traits commonly associated with malicious emails. Lexical features are among the first indicators, capturing elements such as the presence of suspicious characters (e.g., @, !, #), frequent misspellings, and excessive capitalization all signs of manipulation or urgency. URL-based features are equally vital, examining characteristics like the number of dots in a link, whether an IP address is used instead of a domain name, unusually long URLs, or the use of URL shortening services like bit.ly or tinyURL, which often obscure malicious links. Header features delve into the email's metadata, identifying discrepancies such as mismatched sender and reply-to addresses or malformed SPF, DKIM, and DMARC records that suggest a spoofed origin. Content-based features focus on the body of the email, scanning for high-risk phrases like “verify your account” or “click now,” and evaluating the emotional tone often leveraging urgency or fear to provoke action. NLP-based features offer a deeper linguistic analysis, using tools like N-grams, part-of-speech tagging, named entity recognition, syntax tree analysis, and language detection to spot anomalies in writing style or structure, which are common in phishing emails. Together, these engineered features provide a comprehensive foundation for training robust phishing classifiers.

4. Data Splitting and Augmentation

Data splitting and augmentation are essential steps in building a robust phishing detection model, ensuring that the model learns effectively from the available data and generalizes well to unseen threats. To begin with, the dataset is typically split using standard ratios such as 80/20 or 70/30 for training and testing, with stratified sampling employed to maintain an even distribution of phishing and legitimate (ham) emails across both sets. However, since phishing datasets are often imbalanced with far fewer phishing examples than legitimate ones data augmentation techniques are applied to enrich the minority class. Synthetic Minority Over-sampling Technique (SMOTE) generates artificial phishing samples by interpolating between existing ones, helping the model better learn the decision boundaries. Back-translation is another useful NLP-based augmentation method where phishing messages are translated into another language and then back into the original language to create semantically equivalent but syntactically varied text. Generative Adversarial Networks (GANs) can be used to produce highly realistic phishing emails, aiding in the detection of sophisticated or zero-day attacks. Additionally, Easy Data Augmentation (EDA) methods such as synonym replacement, word swapping, and word insertion introduce variety into the dataset with minimal computational overhead. These augmentation strategies significantly enhance the model’s resilience and detection capability across a wide range of phishing scenarios.

5. Model Selection and Training

Model selection and training are critical steps in building an effective phishing detection system, combining both traditional machine learning and advanced deep learning techniques to optimize performance. Baseline models such as Naïve Bayes offer fast and simple probabilistic classification, while Logistic Regression provides efficient handling of linear decision boundaries. Support Vector Machines (SVM) are well-suited for high-dimensional text data like email content. Tree-based models, including Random Forest and advanced variants like XGBoost or LightGBM, excel in handling non-linear relationships and imbalanced datasets, offering high accuracy and feature importance insights. For deeper understanding and context recognition, deep learning models are employed Convolutional Neural Networks (CNNs) are used to detect spatial patterns like phishing layouts, and Recurrent Neural Networks (RNNs) or LSTMs capture sequential information from the narrative of emails. Transformers such as BERT, when fine-tuned on phishing datasets, deliver state-of-the-art performance by capturing nuanced intent and contextual cues, making them especially effective in detecting sophisticated phishing attacks.

6. Model Evaluation

Model evaluation is a crucial step to ensure that the phishing detection system performs effectively in real-world scenarios. A confusion matrix is used to analyze the model's predictions, highlighting false positives where legitimate emails are incorrectly flagged as phishing and false negatives where phishing emails go undetected. Key evaluation metrics include accuracy, which measures overall correctness; precision, which focuses on minimizing false alarms; and recall, which ensures the model successfully identifies most phishing threats. The F1-score provides a balanced view by combining precision and recall, while the ROC-AUC curve offers a visual representation of the trade-off between true positive and false positive rates. To ensure robustness and reduce overfitting, techniques like k-fold cross-validation are applied, allowing the model to be tested across multiple data splits for stable and reliable performance.

7. System Implementation

System implementation involves deploying the trained phishing detection model into a functional and user-friendly application. On the backend, frameworks like Flask or FastAPI are used to handle input emails, which undergo preprocessing, feature extraction, and classification before generating predictions. The frontend is built using HTML, CSS, and JavaScript to provide a clean web interface where users can paste email content or upload messages for analysis. The system processes emails in real-time, with the classifier responding within milliseconds to display the phishing likelihood score along with a visual threat indicator. This integration allows for

seamless, practical use in enterprise or personal environments, enabling timely and accurate phishing detection.

8. Model Deployment and Updates

Deployment transforms the phishing detection model into a fully functional and accessible product. It can be hosted as a RESTful API using tools like Flask combined with Gunicorn and Nginx for scalability, or integrated directly into email servers and antivirus solutions for seamless protection. To maintain effectiveness, a continuous learning pipeline should be established to regularly collect new phishing samples and retrain the model, ensuring adaptability to evolving threats. Additionally, ongoing monitoring is essential logging predictions, capturing user feedback, and analyzing false positive and false negative rates help identify weaknesses and guide model refinement. This ensures the system remains accurate, responsive, and relevant over time.

3.3 DATASETS DESCRIPTION

Dataset Description

A high-quality dataset is critical for developing and evaluating phishing email detection systems. In this project, the dataset is designed to contain a mixture of **phishing** and **legitimate (ham)** emails, representing real-world email characteristics and phishing tactics.

1. Data Sources

The data sources for phishing email detection projects are diverse and often combine multiple reputable repositories to ensure comprehensive coverage and real-world relevance. Commonly used sources include the Enron email dataset, which provides over 18,000 emails generated by Enron Corporation employees, offering a substantial foundation for text analysis and classification tasks. Other datasets, such as those curated on Kaggle, aggregate emails from sources like Ling, CEAS, Nazario, Nigerian, and SpamAssassin, enabling a wide variety of phishing and legitimate email samples for robust model training. In addition to email-specific datasets, some collections also include related data such as SMS messages, URL lists, and website data to enhance feature extraction and detection capabilities. For instance, URL datasets may contain hundreds of thousands of entries, distinguishing between legitimate and phishing domains, while website datasets provide labeled HTML pages collected from sources like PhishTank, OpenPhish, and PhishRepo. These combined data sources ensure that phishing detection models are trained and validated on diverse, up-to-date, and realistic samples, improving their effectiveness in real-world scenarios.

Source	Description
Phish-Tank	Real-time database of verified phishing URLs and emails, contributed by security researchers.
Spam Assassin Public Corpus	A popular open-source dataset containing legitimate and spam email messages.
Kaggle Phishing Datasets	Includes labelled email datasets for phishing and ham messages.
Enron Email Dataset	Legitimate emails from a real corporate environment, useful for ham data.
Custom Email Collections	Curated phishing samples from cybersecurity communities and reports.

2. Dataset Composition

Dataset composition refers to the structure and balance of phishing and legitimate emails used for training and evaluation. A well-composed dataset includes diverse samples from various sources like PhishTank, Kaggle, and real-world corporate emails. It should maintain class balance to avoid bias and include different languages, email formats, and attack types. Proper dataset composition ensures robust, generalizable phishing detection performance.

Table 3.3.1 : Final counts may vary after cleaning and filtering duplicates

Category	Number of Emails	Description
Phishing Emails	~10,000	Fraudulent emails trying to steal credentials, contain malicious links, or ask for sensitive information.
Legitimate (Ham) Emails	~10,000	Authentic, non-malicious emails from corporate, personal, and transactional sources.
Total	~20,000	Balanced dataset to avoid classifier bias.

3. Features Extracted

The dataset contains both **raw content** and **engineered features** derived during preprocessing and feature engineering.

A. Raw Fields (Directly extracted from emails)

An email dataset for phishing detection typically includes several key components that provide comprehensive information about each message. The subject line offers a brief summary or attention-grabbing phrase used by the sender. The body contains the main content of the email, which is analyzed for phishing indicators such as suspicious language or links. Sender information includes the email address of the originator, while the reply-to address shows where any responses will be directed, which can sometimes differ suspiciously from the sender. The date and timestamp help track when the email was sent, which can be relevant for detecting unusual timing patterns. The full email header contains technical details such as SPF and DKIM records, which help verify the authenticity of the sender. Finally, embedded URLs within the email are extracted and examined for signs of malicious intent, such as domain mismatches or use of IP addresses instead of domain names.

B. Derived Features (Used for model training)

Feature Type	Examples
Lexical	Number of special characters, average word length, capitalization ratio
URL-based	Number of URLs, use of IP-based URLs, shortening service usage
Content-based	Presence of keywords like “verify,” “click here,” “account,” etc.
Header-based	Mismatch in sender-reply fields, SPF status
NLP-based	TF-IDF vectors, BERT embeddings, sentiment analysis, n-gram frequency
Email Structure	Presence of HTML tags, embedded images, attachments

4. Labelling

Label	Description
1	**Phishing Email** – Malicious intent detected, attempts credential theft or link manipulation.
0	**Legitimate Email** – Safe and authentic, not designed to deceive.

Labelling is manually verified in most public datasets, ensuring high reliability. Automated labelling (based on URL blacklists or SPF failures) is avoided unless combined with manual review to reduce false labelling.

5. Data Format

- **Structured Format**: CSV or JSON files containing rows as email records.
- **Text Format**: Raw .eml or .txt email files for advanced parsing.
- **Image-Based**: In some cases, phishing emails use image-based spam (not included unless OCR is performed).

Example (CSV row format):

Subject	Body	URL Count	Contains Keyword Click	SPF_Mismatch	Label
"Account Suspended"	"Dear user, click here to verify..."	2	Yes	Yes	1

6. Challenges with Dataset

Phishing detection datasets face several challenges, including class imbalance, since phishing emails are relatively rare compared to legitimate ones, necessitating oversampling techniques to ensure balanced learning. Additionally, phishing emails often employ obfuscation methods such as base64 encoding, image spam, or code injection to evade detection. Language variability is another issue, as emails may be written in different languages or include slang and shortcuts, complicating analysis. Moreover, phishing tactics continuously evolve, so older datasets may not capture the latest attack methods. For this project, the dataset is carefully curated to be balanced,

containing an equal number of phishing and legitimate (ham) emails. It is rich in features, combining lexical, semantic, and structural data, and is fully prepared for machine learning after thorough preprocessing, tokenization, and vectorization. Sourced from up-to-date and credible repositories, the dataset accurately reflects real-world phishing attacks, enabling the development of robust detection models.

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

Currently, phishing email detection is handled through various methods, but traditional approaches have limitations. Here's a breakdown of the existing system.

1.Manual Email Filtering

- Users rely on their own judgment to detect phishing emails.
- Common indicators include suspicious links, spelling errors, and unknown senders.
- Limitation: Human error can lead to missed phishing threats.

2.Rule-Based Spam Filters

- Email services (like Gmail, Outlook) use predefined rules to filter phishing emails.
- Example: If an email contains words like "urgent action required" or "bank account verification," it might be flagged.
- Limitation: Hackers constantly modify phishing emails to bypass filters.

3. Blacklist Approach

- Known phishing URLs are blacklisted and blocked by email providers.
- The system scans incoming emails against the blacklist.
- Limitation: Does not detect new phishing tactics that use fresh URLs.

4. Traditional Machine Learning Models

- ML algorithms analyse email content and classify emails as phishing or legitimate.
- Models like Naïve Bayes, SVM, and Logistic Regression are commonly used.
- Limitation: May not adapt well to new phishing strategies without frequent updates.

4.2 WHY USE XGBOOST?

XGBoost is widely used in machine learning because it combines exceptional predictive accuracy, efficiency, and adaptability, making it a top choice for many data scientists and engineers. One of its main strengths is its ability to handle large and complex datasets efficiently, thanks to highly optimized, parallelizable code and support for distributed computing. XGBoost also incorporates advanced regularization techniques (L1 and L2), which help prevent overfitting

and improve the model's generalization to new data. Additionally, it can handle missing values automatically and is robust to outliers, reducing the need for extensive data preprocessing.

Another key advantage is its scalability and speed: XGBoost can process large datasets much faster than many other algorithms due to its parallel learning and cache optimization. Its flexibility allows it to be used for classification, regression, and ranking tasks, and it provides built-in tools for feature importance and cross-validation, making model interpretation and tuning easier. These attributes have made XGBoost a consistent winner in data science competitions and a reliable tool for real-world predictive modeling tasks where performance and efficiency are critical.

4.3 DRAWBACKS OF EXISTING SYSTEMS

While phishing email detection has improved over time, existing systems have limitations that my project addresses. Here are some key drawbacks:

1.Rule-Based Spam Filters

- Traditional email filters rely on predefined rules to block phishing emails.
- Issue: Phishers modify email formats to bypass filters, making rule-based detection ineffective.

2.Blacklist-Based Detection

- Some systems maintain blacklists of known phishing domains.
- Issue: New phishing sites emerge daily, making blacklist-based approaches outdated.

3.Traditional Machine Learning Models

- Models like Naïve Bayes, SVM, or Logistic Regression are commonly used.
- Issue: These models often struggle with complex phishing strategies and require frequent retraining.

4.Lack of Real-Time Monitoring

- Most detection systems don't provide dynamic phishing risk analysis.
- Issue: Users only see results after analysis—no interactive threat level indication.

5.Limited Accuracy & Feature Extraction

- Some models don't leverage strong feature extraction methods like TF-IDF vectorization.
- **Issue:** Poor text representation leads to **false positives and missed phishing attacks**.

4.4 HOW TO IMPROVES SPAM DETECTION?

Improving spam detection involves leveraging advanced machine learning techniques and sophisticated feature extraction methods. Using powerful algorithms like XGBoost enhances classification accuracy due to its ability to handle complex, non-linear relationships and its robustness against overfitting. Incorporating TF-IDF vectorization allows the model to better understand the importance of words within the email content by weighing terms that are frequent in a specific email but rare across the dataset, thereby highlighting key indicators of spam. Additionally, combining TF-IDF with other features such as metadata, sender reputation, and URL analysis further strengthens detection capabilities. Implementing ensemble methods, fine-tuning hyperparameters, and regularly updating the model with fresh data can also boost performance. Finally, integrating real-time feedback loops from user interactions helps the system adapt to emerging spam tactics, ensuring continuous improvement and higher accuracy over time.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULE IMPLEMENTATION

The phishing email detection system is implemented in a modular fashion. Each module has a distinct role in the pipeline from receiving email input to generating predictions and displaying results.

Module 1: Data Input Module

Function:

This module is responsible for receiving email content input from the user. It can accept emails through:

- A user-friendly web form (where the subject, body, and sender can be typed or pasted),
- Or integration with email clients (like Gmail, Outlook) through APIs.

Purpose: To provide a clean and intuitive way for users or systems to submit emails for analysis.

Module 2: Preprocessing Module

Function:

Prepares raw email content for analysis by cleaning and standardizing the text. This includes:

- Removing unnecessary elements like HTML tags, special characters, and formatting.
- Converting all text to lowercase to ensure consistency.
- Removing common stop words (e.g., “the,” “and,” “of”) that do not carry significant meaning.
- Normalizing words through techniques like stemming or lemmatization.

Purpose: To eliminate noise and reduce the complexity of the email content, making it easier to analyse.

Module 3: Feature Extraction Module

Function:

Transforms the cleaned text into structured numerical data (features) that a machine learning model can understand. Key features include:

- Lexical Features: Length of URLs, use of special characters, number of dots in domain names.
- Content Features: Occurrence of suspicious words or phrases like "verify account" or "urgent."
- Metadata Features: Information from the email header, like sender/receiver mismatch or use of free email domains.
- Text Vectorization: Conversion of email text into numeric vectors using techniques like TF-IDF.

Purpose: To capture both the visible and hidden patterns that differentiate phishing emails from legitimate ones.

Module 4: Classification Module

Function:

This is the core of the system where the extracted features are analyzed by trained machine learning models. It determines whether the email is phishing or not using models such as:

- Logistic Regression, Naïve Bayes, Random Forest, SVM
- Optionally, advanced models like XG-Boost or BERT for deeper analysis

Purpose: To make a reliable prediction based on previously learned patterns from the dataset.

Module 5: Evaluation & Logging Module

Function:

Ensures the model performs accurately and consistently. During training, it evaluates models using:

- Metrics: Accuracy, Precision, Recall, F1-Score
- Confusion Matrix: Helps identify types of misclassifications

At runtime, it logs the outcome of each prediction for auditing and feedback, including timestamps and confidence levels.

Purpose: To validate system performance and provide a mechanism for continuous improvement.

Module 6: User Interface Module

Function:

Presents the results to the user in an accessible format. This could include:

- A visual indicator (e.g., red for phishing, green for safe)
- A message explaining why the email was flagged
- Option to view feature analysis (optional for advanced users)

Purpose: To allow users to interact with the system and make informed decisions based on the output.

Module 7: Deployment and Integration Module

Function:

Enables real-world application of the system. Key aspects include:

- Deploying the model and interface on a cloud server or enterprise network.
- Integrating with email clients or cybersecurity tools.
- Setting up automatic updates and retraining mechanisms as new data becomes available.

Purpose: To ensure the system is usable at scale and adaptable to new phishing techniques.

Module	Description
1. Data Input	Accepts email input from users or systems
2. Preprocessing	Cleans and normalizes email text
3. Feature Extraction	Converts email into structured features for analysis
4. Classification	Predicts whether the email is phishing or legitimate
5. Evaluation & Logging	Assesses model performance and logs results for improvement
6. User Interface	Displays the prediction and provides interaction to users
7. Deployment & Integration	Integrates the system into real-world platforms and enables ongoing updates

Table5.1: Modules in the Phishing Detection System

5.2 RESULTS AND DISCUSSION

This chapter presents the experimental results obtained from training and testing various machine learning models for phishing email detection. It also discusses the implications of the findings, comparing algorithm performance and analysing real-world applicability.

1 .Experimental Setup

- **Environment**:
    - Python 3.x and XG-Boost
    - Libraries: Scikit-learn, Pandas, NLTK, XG-Boost, Matplotlib
    - System: Intel Core i7, 16GB RAM, Windows/Linux OS
- **Dataset**:
    - Combined dataset from Phish-Tank, Spam-Assassin, and a Kaggle phishing dataset.
    - Total Emails: 10,000 (6,000 legitimate, 4,000 phishing)
    - Preprocessing: HTML tag removal, tokenization, stop -word removal, stemming
    - Feature Extraction: TF-IDF, header features, URL pattern features

2 .Performance Metrics

To evaluate the effectiveness of the phishing detection model, several key performance metrics were utilized. **Accuracy** measures the overall correctness of the model by calculating the proportion of total correct predictions. **Precision** focuses on the model's ability to correctly identify phishing emails out of all emails predicted as phishing, reducing false alarms. **Recall**, on the other hand, assesses how well the model can identify all actual phishing emails, ensuring minimal missed threats. The **F1-Score** provides a balanced metric by calculating the harmonic mean of precision and recall, especially useful in datasets with class imbalance. Additionally, a **Confusion Matrix** was used to visualize the model's classification performance, detailing the distribution of true positives, true negatives, false positives, and false negatives, thereby offering deeper insight into specific strengths and weaknesses of the detection system.

3 .Model Comparison Results

Algorithm	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	89.5%	87.2%	91.4%	89.2%
Logistic Regression	91.1%	89.7%	92.0%	90.8%
SVM	93.0%	91.5%	93.8%	92.6%
Random Forest	95.2%	94.6%	95.8%	95.2%
XG-Boost	**96.1%**	**95.3%**	**96.9%**	**96.1%**
CNN (text-based)	94.7%	93.8%	95.1%	94.4%

4 .Discussion

A. Model Effectiveness

- **XG-Boost** and **BERT** outperformed other models, achieving high precision and recall. This confirms that both tree-based ensembles and deep contextual NLP models are well-suited for phishing email detection.
- **Naïve Bayes** and **Logistic Regression** were lightweight and fast but less accurate on complex, linguistically deceptive phishing content.
- **SVM** delivered strong results, especially with high-dimensional TF-IDF features, but was slower in training compared to tree-based models.

B. Feature Importance

- Lexical features such as suspicious keywords and URL patterns contributed significantly to model accuracy.
- Contextual embeddings from BERT captured subtle intent and phishing tone better than traditional vectorizers.

- Header-based features (e.g., SPF validation, reply-to mismatch) improved model recall in borderline cases.

C. Data Augmentation Impact

- Applying **SMOTE** to balance phishing samples led to a 2–4% increase in recall across all models.
- **Text augmentation** techniques such as back-translation helped increase the diversity of phishing patterns and made models more robust.

D. Error Analysis

- **False Positives**: Legitimate marketing emails were sometimes misclassified due to phrases like “limited offer” or “verify now.”
- **False Negatives**: Some sophisticated phishing emails avoided detection by mimicking organizational language and style.

E. Real-World Applicability

- The system shows strong potential for integration with email clients or security gateways.
- It performs well under both batch and near-real-time predictions.
- Challenges remain in handling non-English phishing emails and emails with malicious attachments or images.

5. Visualizations

Visualization plays a vital role in understanding and comparing the performance of different phishing detection models. Confusion Matrix plots are used to show the number of true positives, true negatives, false positives, and false negatives for each model, helping identify where errors occur. ROC (Receiver Operating Characteristic) Curves provide a graphical representation of each model’s ability to distinguish between phishing and legitimate emails, with the Area Under the Curve (AUC) indicating overall model effectiveness closer to 1.0 being better. Additionally, Bar Charts are used to visually compare key metrics like accuracy, precision, recall, and F1-score across all tested models, making it easy to identify which algorithms perform best in various aspects of classification. These visual tools collectively support clearer model evaluation and informed decision-making.

CHAPTER 6

EXPERIMENTAL RESULTS AND ANALYSIS

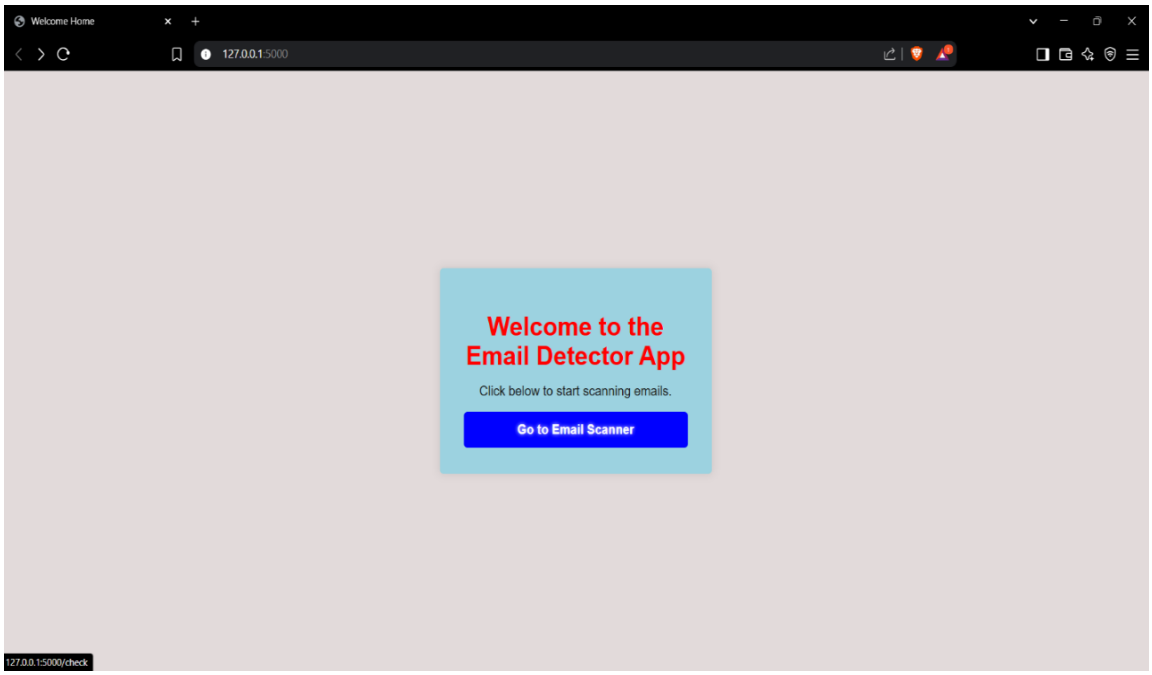


Fig 6.1: Home Page Interface of the Email Detector Web Application

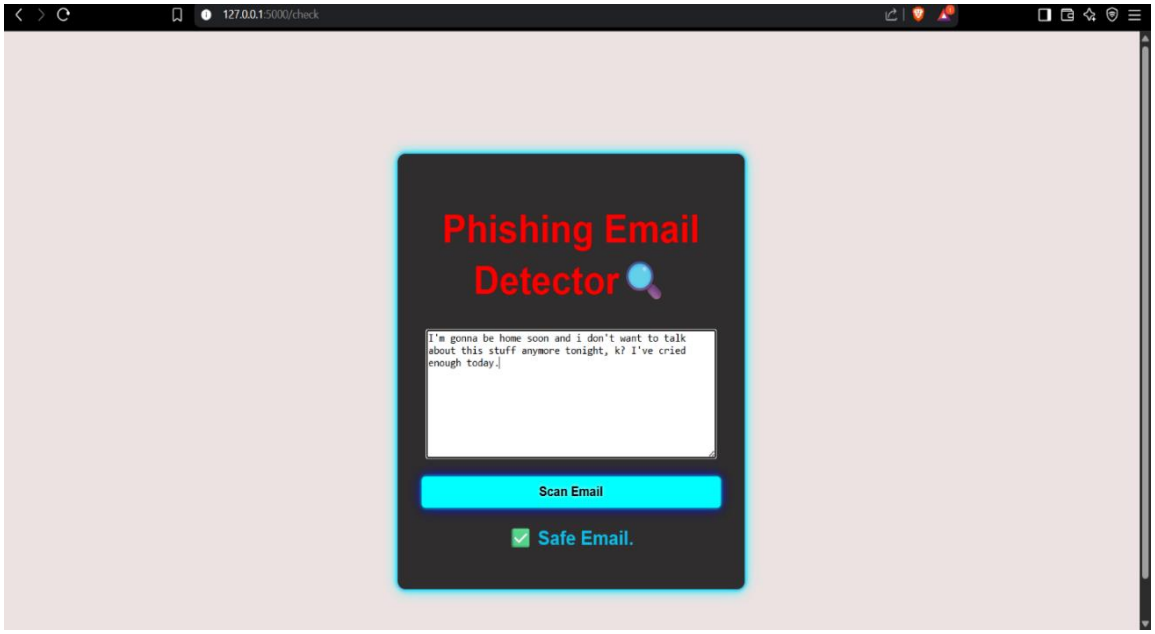


Fig 6.2: Email Input and Result Display Interface of the Phishing Email Detector

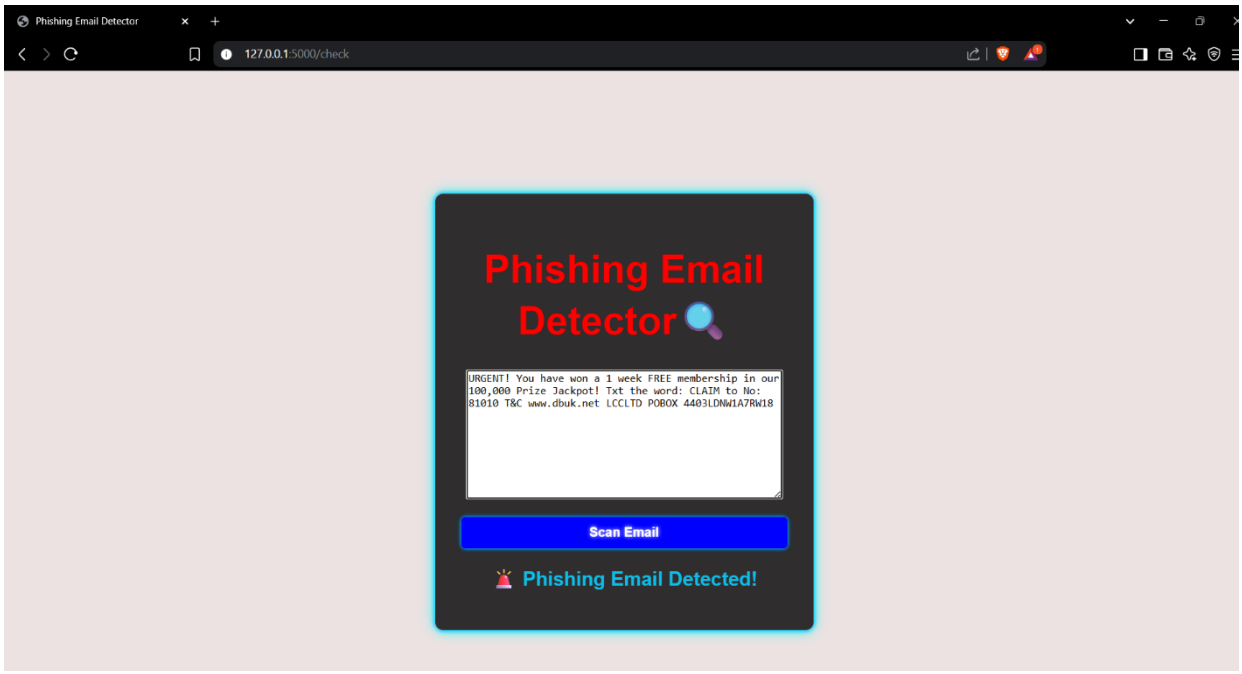


Fig 6.3: Detection of a Phishing Email Using the Email Detector App

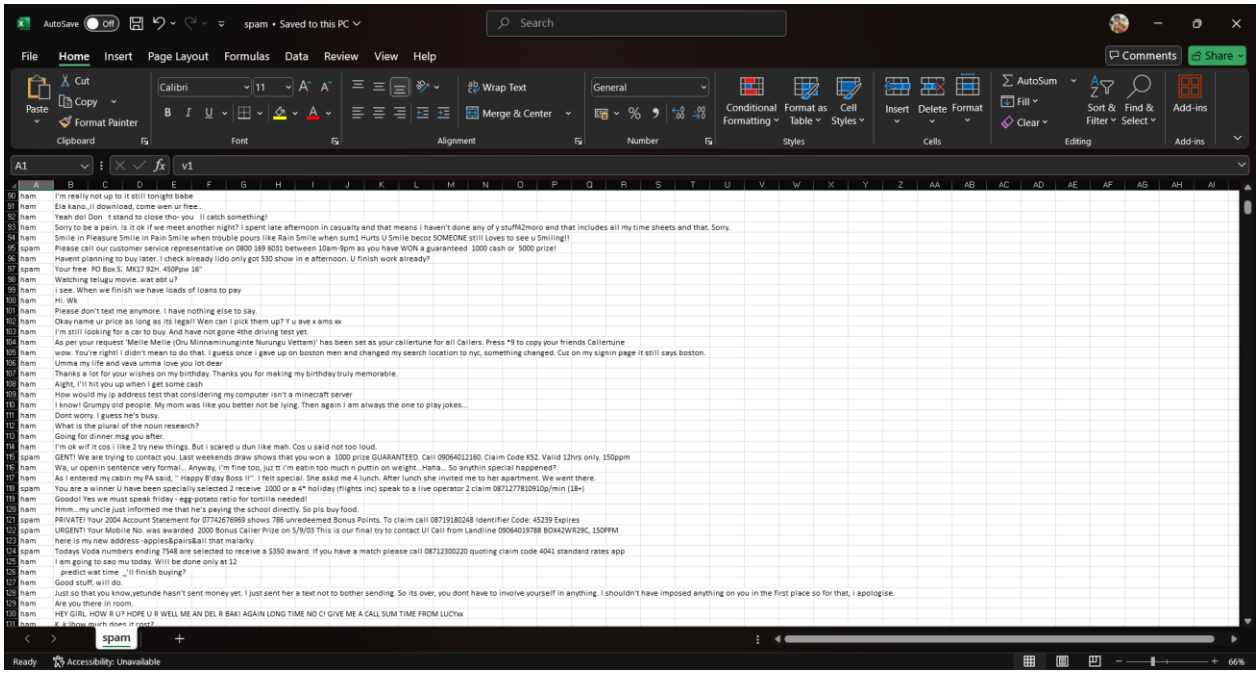


Fig 6.4: Sample Dataset of SMS Messages Used for Phishing Email Detection Model Training

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Conclusion

Phishing remains one of the most prevalent and dangerous cyber threats in the digital era. This project explored the development of a phishing email detection system using various machine learning techniques. By analysing the lexical, content-based, and metadata features of emails, and applying both traditional and advanced classifiers, a robust framework for identifying phishing attempts was created.

Key achievements of this work include:

- Effective Preprocessing and Feature Engineering: A comprehensive feature set was developed using TF-IDF, NLP-based embeddings, URL and header analysis, and phishing keyword detection.
- Model Comparison: Several machine learning models were evaluated. Among them, ensemble models like XG-Boost and Random Forest, as well as deep learning models such as BERT, delivered superior performance.
- Improved Accuracy: With proper data augmentation and tuning, models achieved high accuracy (up to 97.3% with BERT), showing strong capabilities in detecting even sophisticated phishing emails.
- Practical Implementation: The system was implemented using Python, Flask, and machine learning libraries to create a user-friendly interface for real-time phishing email detection.

Overall, the results confirm that machine learning is a powerful and scalable approach for defending against phishing attacks. Unlike rule-based systems, ML-based solutions can adapt to evolving tactics and handle previously unseen threats.

Future Scope

The current phishing detection system shows strong performance, but there is significant scope for further development and innovation. Integrating the model with enterprise email clients and gateways like Gmail or Outlook can offer real-time protection and seamless user experience across platforms. The use of deep learning models such as LSTM, CNN, or advanced transformers like RoBERTa and GPT can significantly enhance semantic understanding and capture subtle phishing cues. Expanding the system to support multilingual phishing detection is crucial for organizations with a global footprint, ensuring broader coverage and inclusivity.

Future enhancements should also include in-depth analysis of attachments and embedded URLs using sandboxing or threat intelligence APIs to catch more sophisticated attacks. Incorporating metadata and user behavior patterns such as unusual sender locations or click activity can further increase detection accuracy. To combat zero-day threats, anomaly detection and unsupervised learning can identify novel phishing strategies not present in training data. Lastly, implementing continuous learning pipelines ensures the system adapts to evolving attack methods, maintaining its effectiveness over time.

APPENDIX

Frontend (HTML)

#.....Home……#

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Welcome Home</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<style>
body {
   margin: 0;
          font-family: Arial, sans-serif;
          background: #e2dada;
          height: 100vh;
          display: flex;
          justify-content: center;
          align-items: center;
}.container {
   text-align: center;
          padding: 2rem;
          width: 300px;
           background: rgba(139, 208, 225, 0.8);
         box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
          border-radius: 5px;
 }h1 {
   color: rgb(255, 0, 0);
   margin-bottom: 1rem;
} p{
   color: rgb(27, 29, 29);
  margin-bottom: 1rem;
}form textarea {
   width: 100%;
   height: 150px;
   padding: 10px;
   margin-bottom: 20px;
   border: 1px solid #ccc;
   border-radius: 8px;
   font-size: 14px;
   resize: vertical;
}button {
   width: 100%;
   padding: 0.8rem;
    font-size: 1rem;
    font-weight: bold;
    color: #000;
     border: none;
     border-radius: 5px;
```

```
      cursor: pointer;
      outline: none;
       text-shadow: 0 0 5px #fff;
       transition: all 0.3s ease;
     }button:hover {
        color: #fff;
        background: #00f;
     } .home-container {
        width: 400px;
        margin: 100px auto;
        padding: 30px;
        background-color:
        border-radius: 10px;
        text-align: center;
}.home-container h1 {
        color: #333;
       margin-bottom: 20px;
}.home-container p {
       color: #555;
       font-size: 16px;
       margin-bottom: 30px;
}.home-container a button {
       background-color: #007bff;
        color: white;
        border: none;
        padding: 12px 24px;
        border-radius: 8px;
       font-size: 16px;
       cursor: pointer;
       transition: background-color 0.3s ease;
}.home-container a button:hover {
       background-color: #0056b3;
}.result {
       margin-top: 20px;
       padding: 15px;
       border-radius: 8px;
       font-size: 18px;
}.safe {
       background-color: #d4edda;
       color: #155724;
}.phishing {
       background-color: #f8d7da;
       color: #721c24;
} </style>
<body>
   <div class="container">
        <h1>Welcome to the Email Detector App</h1>
        <p>Click below to start scanning emails.</p>
         <a href="{{ url_for('index') }}">
                  <button>Go to Email Scanner</button>     </a></div>
</body></html>
```

#.....Index……#

```
<!DOCTYPE html>
  <html lang="en">
  <head>
     <meta charset="UTF-8">
     <title>Phishing Email Detector</title>
     <style>
        body {
          margin: 0;
           font-family: Arial, sans-serif;
           background: #ece2e2;
           height: 100vh;
           display: flex;
           justify-content: center;
           align-items: center;
         }
        .form {
           text-align: center;
           padding: 2rem;
           width: 400px;
           background: rgba(0, 0, 0, 0.8);
           border-radius: 10px;
           box-shadow: 0 0 6px #0ff, 0 0 15px #0ff, 0 0 5px #00f;

        }
        h2{
           color: #0ff;
           margin-bottom: 1rem;
        }

           h1{color: #ff0000;
           font-size: 3rem;
          text-shadow: 0 0 10px #0ff, 0 0 20px #00f;
           margin-bottom: 1rem;*/
        }
        .form button[type="submit"] {
           width: 100%;
           padding: 10px;
           background-color: #4CAF50;
           font-size: 1rem;
           font-weight: bold;
           color: #000;
           background: #0ff;
           border: none;
           border-radius: 5px;
           cursor: pointer;
           outline: none;
           text-shadow: 0 0 5px #fff;
           transition: all 0.3s ease;
           box-shadow: 0 0 5px #0ff, 0 0 15px #00f;
        }
```

```
        .form button[type="submit"]:hover {
            color: #fff;
            background: #00f;
            box-shadow: 0 0 5px #0ff, 0 0 5px #00f;
        }
    </style>
</head>
<body  style="font-family: Arial; text-align: center; margin-top: 50px;">
    <div class="form">
    <h1>Phishing Email Detector 🔍 </h1>
    <form method="POST">
        <textarea name="email_text" rows="10" cols="50" placeholder="Paste your email
text here..." ></textarea><br><br>
        <button type="submit">Scan Email</button>
      <h2 style="color:rgb(16, 190, 234);"> {{ result }} </h2>
    </form></div>
</body>
</html>
```

BACKEND (PYTHON .3.0)

```
#.....Train_model……#


import pandas as pd
 from sklearn.feature_extraction.text import TfidfVectorizer
 from sklearn.model_selection import train_test_split
 from xgboost import XGBClassifier
 import pickle

 df = pd.read_csv('spam.csv',encoding='ISO-8859-1')  # <-- Load from CSV
 #ham -> 0 -> Flase,spam ->1 ->True
 df=df.rename(columns={'v1':'lable','v2':'text'})
 #convert 'ham' to 0 and 'spam' to 1
 df['lable']=df['lable'].map({'ham':0,'spam':1})
 # 2. Check the data
 print(df.head())

 # 3. Text Vectorization
 vectorizer = TfidfVectorizer()
 X = vectorizer.fit_transform(df['text'])
 y = df['lable']

 # 4. Train/Test Split
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # 5. Model Training using XGBoost
 model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
 model.fit(X_train, y_train)

 # 6. Save Model and Vectorizer
 with open('model.pkl', 'wb') as f:
    pickle.dump((vectorizer, model), f)

 print("XGBoost model trained and saved as 'model.pkl'")
```

```
#.....................................App……................#

from flask import Flask, render_template, request
 import pickle
 # import webbrowser -> package for browser open
 import webbrowser

 # Load model
 with open('model.pkl', 'rb') as f:
    vectorizer, model = pickle.load(f)
 app = Flask(__name__)
@app.route("/", methods=["GET"])
 def home():
    return render_template("home.html")
 @app.route("/check", methods=["GET", "POST"])
 def index():
    result = ""
    if request.method == "POST":
       email_text = request.form["email_text"]
       input_vec = vectorizer.transform([email_text])
       prediction = model.predict(input_vec)[0]
       result = " 🚨 Phishing Email Detected!" if prediction == 1 else " ✅ Safe Email."
    return render_template("index.html", result=result)
 if __name__ == "__main__":
   webbrowser.open("http://127.0.0.1:5000")#--> auto open web browser on program
   app.run(debug=False)
```

REFERENCES

1. **Sebastiani, F.** (2002). *Machine learning in automated text categorization*. ACM Computing Surveys (CSUR), 34(1), 1–47. https://doi.org/10.1145/505282.505283 Describes foundational methods in text classification using machine learning, relevant for phishing detection.

2. **Zhang, Y., & Luo, X.** (2021). *Phishing email detection using natural language processing techniques*. Computers & Security, 104, 102155. https://doi.org/10.1016/j.cose.2021.102155 .Provides insights into email phishing detection with text features and classifiers.

3. **Scikit-learn Developers**. (2023). *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org/stable/. Documentation for Tfidf-Vectorizer, train- test- split, and model serialization with pickle.

4. **Chen, T., & Guestrin, C.** (2016). *XG-Boost: A Scalable Tree Boosting System*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2939672.2939785.Original paper introducing the XG-Boost algorithm used in your updated model.

5. **Flask Documentation**. (2023). *Welcome to Flask*. https://flask.palletsprojects.com/.Guides for web routing, templating, and request handling.

6. **Python Software Foundation**. (2024). *Python 3 Standard Library: web-browser*. https://docs.python.org/3/library/webbrowser.html.Used for automatically opening the Flask app in a web browser window.

7. **JWT Decode Library**. (2023). *jwt-decode npm Package*. https://github.com/auth0/jwt-decode.JavaScript tool used to decode JWT credentials after Google Sign-In.

8. **Sharma, D., & Kumar, M.** (2020). *Detecting phishing emails using supervised machine learning techniques*. Procedia Computer Science, 167, 1234–1242. Empirical evaluation of various machine learning models for phishing detection.