**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

ARTIFICIAL INTELLIGENCE APPLICATIONS

IN PHISHING EMAIL CLASSIFICATION

by

Pham Trong Kha

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

© Copyright by Pham Trong Kha 2024

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

ARTIFICIAL INTELLIGENCE APPLICATIONS

IN PHISHING EMAIL CLASSIFICATION

by

Pham Trong Kha

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

Supervisor:

Dr. Le Thanh Hai

© Copyright by Pham Trong Kha 2024

Artificial Intelligence Applications in Phishing Email Classification

Pham Trong Kha

Degree Master of Software Engineering

FPT University

2024

Abstract

With the increasing sophistication of phishing attacks, the need for advanced security measures in email communication has become critical. Traditional rule-based systems are often inadequate in detecting complex phishing patterns, prompting the integration of Artificial Intelligence (AI) for more accurate detection. This thesis presents the development of a web-based application designed for system administrators, featuring AI integration for phishing email classification.

Leveraging a dataset of phishing and legitimate emails from Kaggle, the system applies various machine learning models, including Naive Bayes, Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU), to enhance email security. The architecture incorporates a frontend built with HTML, CSS, and JavaScript, and a backend using Python’s Flask framework, interfacing with a MySQL database for email management.

The system is containerized using Docker to ensure scalability and portability. Key AI models are trained using TensorFlow and deployed to classify incoming emails. The application integrates Microsoft Azure Active Directory for secure user authentication, while real-time phishing alerts are provided through email analysis.

By evaluating model performance through accuracy, precision, recall, and F1-score, the system provides a robust solution for detecting phishing emails in real-time. This research demonstrates the practicality of AI-powered solutions in enhancing email security and offers a scalable tool for system administrators to combat phishing threats.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Le Thanh Hai, for his invaluable guidance throughout this project. I am also thankful to my classmates for their helpful advice and support. A special thanks to my wife and my company for their unwavering support and for providing the necessary resources and encouragement to complete this work.

I am also grateful for the knowledge gained from other courses, which contributed to the success of this thesis. Lastly, I would like to thank FPT University for providing an excellent learning environment and offering courses that are closely aligned with the subject matter of my research.

Table of Contents

[Abstract](#_Toc186617063)

[Acknowledgments](#_Toc186617064)

[List of Tables](#_Toc186617065)

[List of Figures](#_Toc186617066)

[List of Abbreviations](#_Toc186617067)

[Chapter 1: Introduction 1](#_Toc186617068)

[1.1 Motivation 1](#_Toc186617069)

[1.2 Problem Statement 2](#_Toc186617070)

[1.3 Research Objectives 3](#_Toc186617071)

[1.4 Research Scope and Limitations 4](#_Toc186617072)

[1.5 Expected Contributions 6](#_Toc186617073)

[1.6 Thesis Organization 7](#_Toc186617074)

[1.7 Preliminary Timeline 8](#_Toc186617075)

[Chapter 2: Literature Review 9](#_Toc186617076)

[2.1 Introduction to Phishing Detection Methods 9](#_Toc186617078)

[2.1.1 User Awareness Methods 9](#_Toc186617079)

[2.1.2 Software-Based Detection Methods 9](#_Toc186617080)

[2.2 Analysis of Previous Studies 11](#_Toc186617081)

[2.2.1 Traditional Methods 11](#_Toc186617082)

[2.2.2 AI-Based Phishing Detection 12](#_Toc186617083)

[2.2.3 Research Gap 19](#_Toc186617084)

[Chapter 3: Methodology 21](#_Toc186617085)

[3.1 Proposed approach 21](#_Toc186617087)

[3.2 System Architecture 21](#_Toc186617088)

[3.3 Dataset 23](#_Toc186617089)

[3.4 Implementation of Naive Bayes Model 31](#_Toc186617090)

[3.5 Implementation of GRU Model 35](#_Toc186617091)

[3.6 Implementation of LSTM Model 40](#_Toc186617092)

[3.7 Frontend Implementation for Phishing Detection System 44](#_Toc186617093)

[3.8 Azure Graph API and Webhook Intergration 48](#_Toc186617094)

[3.9 Backend Implementation 51](#_Toc186617095)

[Chapter 4: Results and Discussion 56](#_Toc186617096)

[4.1 Results 56](#_Toc186617098)

[4.1.1 Model Evaluation 56](#_Toc186617099)

[4.1.2 Front End and Client Integration Result 59](#_Toc186617100)

[4.2 Discussion 63](#_Toc186617101)

[Chapter 5: Conclusion and Future Work 65](#_Toc186617103)

[5.1 Conclusion 65](#_Toc186617105)

[5.2 Future Work 65](#_Toc186617106)

[References 67](#_Toc186617107)

# List of Tables

Table 1: Preliminary timeline for the Thesis

Table 2: Traditional Phishing detection tools

Table 3: Sample data in dataset.

Table 4: Detailed preprocessing steps

Table 5: Sample email after processing

Table 6: Statistical Summary of Email Text Lengths

Table 7: Performance Comparison: LSTM vs. GRU

Table 8: Naïve Bayes Model Classification

Table 9: GRU Model Classification

Table 10: LSTM Model Classification

Table 11: Comparative Table of Naive Bayes, GRU, LSTM

# List of Figures

Figure 1: Impact of phishing email to organization

Figure 2: Phishing Email Attack Process

Figure 3: Overview Architecture of the Web-Based Phishing Detection System

Figure 4: Phishing Detection Methods

Figure 5: GRU Architecture

Figure 6: LSTM Architecture

Figure 7: Detail Architecture of the Web-Based Phishing Detection System

Figure 8: Web Application for Phishing Email Dectection

Figure 9: Dataset summary after balancing

Figure 10: Wordcloud before preprocessing data

Figure 11: Wordcloud after preprocessing data

Figure 12: Data splitting into Training, Validation and Test sets

Figure 13: Top 20 Words by TF-IDF Score

Figure 14: Confusion Matrix and Classification report

Figure 15: GRU Model Summary

Figure 16: GRU Confusion Matrix and Classification report

Figure 17: GRU and LSTM architectures

Figure 18: LSTM Model Summary

Figure 19: LSTM Loss and Accuracy over epochs

Figure 20: LSTM Confusion Matrix and Classification report

Figure 21: Email Protection Center dashboard page

Figure 22: Email Protection Center detail page

# List of Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form** |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| BERT | Bidirectional Encoder Representations from Transformers |
| GRU | Gated Recurrent Unit |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| SQL | Structured Query Language |
| URL | Uniform Resource Locator |
| OAuth | Open Authorization |
| JSON | JavaScript Object Notation |
| HTTPS | Hypertext Transfer Protocol Secure |
| CSV | Comma-Separated Values |
| SMTP | Simple Mail Transfer Protocol |
| IMAP | Internet Message Access Protocol |
| RNN | Recurrent Neural Network |
| NLP | Natural Language Processing |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |
| GPU | Graphics Processing Unit |
| CPU | Central Processing Unit |
| AWS | Amazon Web Services |
| GDPR | General Data Protection Regulation |
| mBERT | Multilingual Bidirectional Encoder Representations from Transformers |
| SMTP | Simple Mail Transfer Protocol |
| MIME | Multipurpose Internet Mail Extensions |
| OAuth | Open Authorization |
| SVM | Support Vector Machine |

# Chapter 1: Introduction

## Motivation

Email is one of the most widely used communication tools for both personal and professional exchanges. However, it has also become a major vector for cyberattacks, particularly phishing attacks. Phishing is a type of cybercrime in which attackers deceive individuals into revealing sensitive information, often leading to data breaches and financial losses. In 2023, it was reported that phishing emails accounted for over 90% of global cyberattacks, making it one of the most pressing cybersecurity issues today [1].

Traditional methods for detecting phishing emails, such as rule-based filtering, are often insufficient in dealing with the sophisticated nature of modern phishing techniques. These systems rely on static rules or keyword matching, which attackers can easily bypass by altering their email patterns. This leads to high false-positive rates, where legitimate emails are incorrectly flagged, and, worse, some phishing emails may go undetected, putting organizations at significant risk [2]. As phishing techniques evolve, there is an increasing need for more advanced detection mechanisms that can adapt to new types of attacks.

Recent advancements in Artificial Intelligence (AI) and Machine Learning (ML) have opened up new possibilities for addressing these challenges. AI, particularly deep learning models like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), has shown considerable promise in improving phishing detection by identifying complex patterns and behaviors within email content that traditional methods miss [3]. These models can learn from large datasets, continuously improving their ability to distinguish phishing emails from legitimate ones.

Moreover, the COVID-19 pandemic has accelerated the shift towards digital communication, with remote work becoming the norm. This has resulted in a significant increase in phishing attacks, with reports indicating a 220% rise in phishing activity during the pandemic [4]. Cybercriminals have exploited this shift by targeting individuals and organizations with more personalized and convincing phishing emails, making the need for robust detection systems even more urgent.

Given these developments, this research focuses on building a web-based application that integrates AI models for classifying phishing emails in real-time. By utilizing advanced machine learning models like Naive Bayes, LSTM, and GRU, the proposed system aims to provide an effective solution for system administrators, improving their ability to detect and mitigate phishing attacks. This approach not only addresses the limitations of traditional systems but also offers scalability and adaptability, making it suitable for modern cybersecurity environments.

## Problem Statement

Phishing involves using deceptive methods through computer-based techniques to trick people into sharing sensitive personal information. For instance, an attacker might send an email that appears to come from a trusted organization like an online retailer, a credit card provider, or a bank. This email is designed to mislead recipients into replying and unintentionally revealing personal details. Phishing attacks often rely on creating a sense of urgency or trust to manipulate victims into acting quickly without verifying the source. [2]

Figure 1: Impact of phishing email to organization

Figure 2 is the process of an email phishing attack. The attack begins with an attacker sending a seemingly legitimate phishing email to a user. The email passes through firewalls and email filtering systems such as anti-virus or spam filters, but these defenses may not detect the threat. When user interacts with the email, attacker can gain access to sensitive information such as social security numbers, credit card details, or other financial data. Additional tools such as proxies, data loss prevention (DLP), and intrusion detection/prevention systems (IPS/IDS) can help protect against such attacks, but attackers often find ways to bypass these security measures.

Figure 2: Phishing Email Attack Process

## Research Objectives

The primary goal of this research is to design and implement a scalable web-based application that provides system administrators with real-time phishing email detection using AI models. The Technical Architecture in Figure 2 is the overview of how the frontend, backend, AI models, and real-time detection processes are integrated to form a cohesive phishing detection solution.

**• Develop a web-based system** for phishing email detection that includes a frontend, backend, and a database for managing email data. The system is built using Flask for the backend, with a frontend developed in HTML, CSS, and JavaScript, and MySQL for data management.

**• Implement AI models** such as Naive Bayes, LSTM, and GRU for phishing classification. These models are trained and deployed using TensorFlow, and Docker is used for containerization, ensuring scalability and deployment across different environments.

**• Integrate Microsoft Azure for authentication and email handling:**

Authentication: Azure Active Directory is used to provide secure authentication for system administrators, ensuring that only authorized personnel can access the system.

Email Handling: Azure is also responsible for connecting to the Microsoft Exchange server to retrieve incoming emails. These emails are then processed through the AI models for phishing detection, and the classification results (phishing or legitimate) are updated back into the user’s mailbox for immediate action.

**• Provide real-time email classification:** The system processes incoming emails in real-time, classifying them as phishing or legitimate and immediately alerting system administrators to any phishing threats.

Figure 3: Overview Architecture of the Web-Based Phishing Detection System

## Research Scope and Limitations

Scope of Research

The primary subject of this research is the development of a web-based application for phishing email classification, aimed at enhancing email security for system administrators. The research focuses on the following key areas:

1. **AI-Powered Phishing Detection**:
   * The core of the research is to integrate advanced machine learning and deep learning models for detecting phishing emails. Models such as Naive Bayes, Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) are used to classify emails as either phishing or legitimate.
2. **Real-Time Email Classification**:
   * The system is designed to operate in real-time, retrieving emails from Microsoft Exchange servers via Microsoft Azure, running them through AI models, and immediately updating the classification back into the user’s mailbox. This real-time aspect is crucial for system administrators to mitigate threats as they arise.
3. **Web-Based Application for System Administrators**:
   * The application provides a user-friendly web interface for system administrators, enabling them to monitor email traffic, view phishing detection results, and receive real-time alerts. The frontend is built with HTML, CSS, and JavaScript, and the backend is powered by Python's Flask framework.
4. **Integration of Microsoft Azure**:
   * Microsoft Azure plays a pivotal role by offering secure authentication through Azure Active Directory and managing email retrieval from Microsoft Exchange servers. This ensures that only authorized users can access the system and that emails are processed securely.

Limitations of Research

Despite the extensive scope of the research, certain limitations are acknowledged:

1. **Dataset Dependency**:
   * The research relies on publicly available datasets, such as the phishing email dataset from Kaggle [5]. While this dataset is diverse, it may not capture all types of phishing techniques encountered in real-world environments. Additionally, the accuracy of the models may be influenced by the quality and diversity of the training data.
2. **Model Generalization**:
   * The AI models implemented (Naive Bayes, LSTM, GRU) are trained on the selected dataset and may not generalize well to emails containing novel phishing techniques that are not represented in the training data. Continuous updates and retraining of models will be required as phishing tactics evolve [6].
3. **Real-Time Performance**:
   * While the system is designed to operate in real-time, performance limitations may arise depending on the volume of incoming emails and the computational resources available. Scaling the system to handle large volumes of emails efficiently will require the use of cloud-based resources, such as those provided by Azure or similar platforms [7][8].
4. **Security Considerations**:
   * Although the system leverages Azure Active Directory for secure authentication, potential vulnerabilities may arise if the system is not regularly updated or if other security layers are compromised. Ensuring that the system remains secure against evolving cyber threats will require ongoing maintenance and security updates.
5. **Phishing Detection Focus**:
   * The scope of the research is limited to phishing email detection. Other forms of cyberattacks, such as malware or ransomware delivered through email, are outside the scope of this study. However, the system could be extended in future research to detect a broader range of email-based threats.

## Expected Contributions

The expected contributions of this research will span both practical applications and academic insights. The proposed system will significantly enhance phishing email detection capabilities while contributing to the field of cybersecurity. The key contributions include:

1. **AI-Based Phishing Detection System**:
   * The development of a web-based phishing email detection system that integrates AI and machine learning models such as Naive Bayes, LSTM, and GRU to improve phishing detection accuracy [3]. This system will address the limitations of traditional rule-based email filters by recognizing complex patterns [2]
2. **Real-Time Classification**:
   * A real-time email classification system that processes emails using Microsoft Azure [7] [8], running them through AI models and updating the classification results directly in user mailboxes. This will empower system administrators to take immediate action against phishing threats.
3. **Enhanced Security for Organizations**:
   * The proposed system will offer a secure authentication framework using Azure Active Directory and a user-friendly interface that allows administrators to monitor email security. This will be a vital tool for organizations seeking to mitigate phishing threats.
4. **Contributions to AI Research in Cybersecurity**:
   * This research will contribute to the field of AI-driven phishing detection, particularly in demonstrating how LSTM and GRU models outperform traditional email filters. This contribution will add valuable knowledge to the academic research community and practical cybersecurity measures.

## Thesis Organization

The thesis will be organized into five primary chapters, each addressing a specific aspect of the research:

**Chapter 1: Introduction**

This chapter introduces the background of phishing detection, the rationale for the research, research objectives, and an overview of the methodology.

**Chapter 2: Literature Review**

This chapter reviews existing studies on phishing detection systems, traditional email filters, and the integration of AI models (Naive Bayes, LSTM, GRU) into phishing detection. It also explores real-time email classification and the role of Microsoft Azure in secure email handling.

**Chapter 3: Methodology**

This chapter outlines the research methodology, including data collection, preprocessing steps, model selection (Naive Bayes, LSTM, GRU), system implementation, and real-time detection through Microsoft Azure.

**Chapter 4: Results and Discussion**

This chapter presents the findings, including model performance metrics (accuracy, precision, recall, F1-score) and compares the proposed system’s effectiveness against traditional methods. It also discusses the challenges encountered during the research.

**Chapter 5: Conclusion and Future Work**

The final chapter summarizes the research outcomes and the effectiveness of the AI-based phishing detection system. It also discusses potential areas for future work, such as extending the system to detect other email-based threats (e.g., malware).

## Preliminary Timeline

The preliminary timeline for the project is as follows:

|  |  |  |
| --- | --- | --- |
| **Task** | **Duration** | **Estimated Completion** |
| **Phase 1: Literature Review** | 15 days | 15-Oct-24 |
| **Phase 2: Data Collection and Preprocessing** | 15 days | 30-Oct-24 |
| **Phase 3: Model Development and Training** | 30 days | 01-Dec-24 |
| **Phase 4: System Implementation** | 20 days | 20-Dec-24 |
| **Phase 5: Model Testing and Evaluation** | 10 days | 31-Dec-24 |
| **Final Submission** | 3 days | 02-Jan-25 |

Table 1: Preliminary timeline for the Thesis

# Chapter 2: Literature Review



## Introduction to Phishing Detection Methods

Phishing attacks are among the most pervasive and damaging cyber threats, targeting individuals and organizations to steal sensitive information such as credentials, financial details, or personal data. Various detection techniques have been developed to mitigate these risks, broadly categorized into User Awareness and Software-Based Detection methods.

### User Awareness Methods

User awareness programs aim to educate individuals about phishing threats and teach them how to identify and avoid suspicious communications. Examples include phishing simulation campaigns, interactive training sessions, and guidelines for spotting suspicious emails.

While user awareness has proven effective in reducing successful phishing attempts, its efficacy is limited by human error and the increasing sophistication of phishing tactics. Studies have shown that combining user training with technical solutions yields better overall protection [10].

### Software-Based Detection Methods

Software-based approaches employ automated solutions to detect phishing attempts. These methods are further classified as follows:

**List-Based Detection**

List-based methods maintain a whitelist of trusted domains and a blacklist of known phishing sources. While straightforward and effective for blocking known threats, these methods fail to address newly emerging phishing sites. Research by Xiang et al. highlighted the rapid obsolescence of blacklists, with phishing sites often having lifespans of just a few hours [11].

**Machine Learning-Based Detection**

Machine learning (ML) algorithms analyze features such as email headers, sender behavior, and content patterns to classify phishing attempts. Commonly used ML algorithms include Support Vector Machines (SVM), Random Forests, and Naïve Bayes classifiers. Studies, such as those by Adebowale et al., have demonstrated the adaptability of ML-based systems in identifying previously unseen phishing strategies [12].

**Deep Learning-Based Detection**

Deep learning techniques, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), provide significant advantages in processing unstructured data such as text and URLs. LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) have shown superior accuracy in phishing detection, as demonstrated in experiments by Bahnsen et al. [3]. However, deep learning models require large datasets and substantial computational resources, posing challenges for implementation in resource-constrained environments.

**Heuristic-Based Detection**

Heuristic detection relies on predefined rules to identify phishing indicators, such as irregular domain structures, unusual keywords, and spoofed visual elements. Although fast and computationally inexpensive, heuristic methods are often circumvented by sophisticated phishing schemes that mimic legitimate entities [13].

**Hybrid Methods**

Hybrid approaches integrate multiple techniques, such as combining ML models with heuristic rules, to enhance detection accuracy. This fusion mitigates individual weaknesses, such as the over-reliance of blacklists or the computational demands of deep learning models. A study by Khonji et al. emphasized that hybrid systems achieve better performance than standalone methods in combating phishing attacks [14].

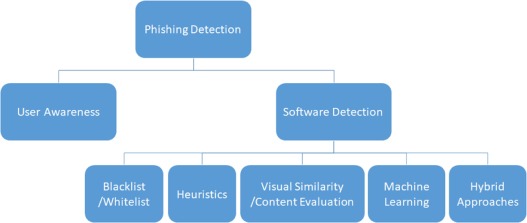


Figure 4: Phishing Detection Methods

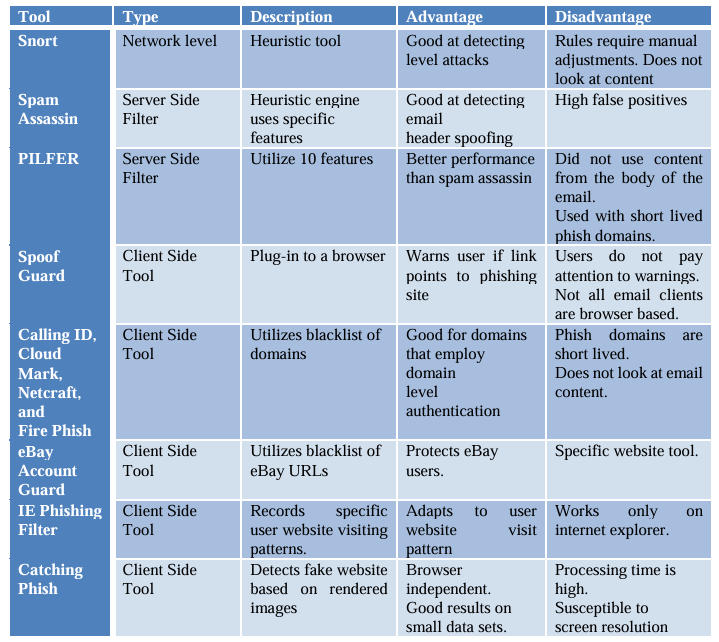


Table 2: Traditional Phishing detection tools

## Analysis of Previous Studies

### Traditional Methods

**Heuristics-Based Approaches**

Heuristics-based phishing detection relies on predefined rules to identify suspicious elements in email messages or embedded URLs. These rules are derived from common characteristics observed in phishing attempts, such as unusual domain names, obfuscated links, or specific keyword patterns (e.g., "verify your account," "urgent action required").

For example, heuristic methods may analyze the structure of URLs to detect embedded IP addresses, excessive subdomains, or unusual TLDs, which are indicative of phishing websites. Patel demonstrated the effectiveness of heuristics in identifying phishing emails by focusing on URL length and entropy features, achieving significant detection rates on a small-scale dataset [9] [13]. Despite its simplicity and computational efficiency, heuristic-based detection is prone to false negatives when attackers craft messages that deviate from known patterns.

**Email Filtering Systems**

Email filtering has been one of the earliest defenses against phishing. Commercial systems like Microsoft Outlook's SmartScreen filter or Gmail's spam detection framework leverage heuristics combined with reputation scoring to block phishing emails before they reach the inbox.

Anti-spam protection in EOP, for example, evaluates sender behavior, domain reputation, and content features to classify emails as safe or potentially malicious [15]. However, while email filters can effectively reduce the volume of phishing emails delivered to users, they are limited by their inability to adapt quickly to new attack vectors. False positives remain a significant concern, as legitimate emails are sometimes flagged erroneously.

**Limitations of Traditional Methods**

Although traditional methods provide a foundational defense against phishing, they are inherently reactive and fail to adapt to evolving phishing tactics. These methods rely heavily on predefined rules, which attackers can bypass by employing sophisticated techniques like domain lookalikes or contextually appropriate wording. Moreover, traditional systems often suffer from scalability issues when applied to large datasets in real-world applications.

### AI-Based Phishing Detection

**Naive Bayes Classifier**

Naive Bayes is a probabilistic machine learning algorithm widely used in phishing detection due to its simplicity and computational efficiency. The model assumes independence between features, allowing it to classify emails based on the likelihood of specific terms appearing in phishing or legitimate emails [16]. For instance, terms like “free” and “offer” might be more prevalent in phishing emails, while “receipt” and “invoice” are common in legitimate ones.

Bayes' Theorem is expressed as:

Where:



* 𝑃(𝐴∣𝐵) is the posterior probability of event A occurring given that B is true.
* P(A∣B) is the likelihood of event B occurring given that A is true.
* P(A) is the prior probability of event A.
* P(B) is the prior probability of event B.

In phishing detection, event A could represent an email being phishing, and event B could represent the presence of specific features within the email.

Assumption of Feature Independence: Naive Bayes assumes that the features used for classification are independent of each other, given the class label. This simplification allows for efficient computation but may not always hold true in real-world scenarios.

Application in Phishing Detection:

* Feature Extraction: Identify relevant features from emails, such as specific words, the presence of links, sender information, and other metadata.
* Training: Calculate the probabilities of each feature occurring in both phishing and legitimate emails using a labeled dataset.
* Classification: For a new email, compute the posterior probability that it is phishing based on the extracted features and classify it accordingly.

Advantages:

* Simplicity: Easy to implement and interpret.
* Efficiency: Requires less computational resources compared to more complex models.
* Performance with Limited Data: Can perform well even with a relatively small dataset.

Limitations:

* Assumption of Independence: The assumption that features are independent may not hold true, potentially affecting accuracy.
* Sensitivity to Feature Selection: The choice of features significantly impacts performance.

Performance Metrics:

Studies have shown varying accuracy levels for Naive Bayes in phishing detection tasks. For instance, one study reported an accuracy of approximately 96% when using Naive Bayes for detecting phishing websites [17]. Another study achieved an accuracy of 97.08% by combining Naive Bayes with ensemble methods like Stacking, Bagging, and Boosting [18].

Comparative Analysis:

When compared to other machine learning algorithms, Naive Bayes may exhibit slightly lower accuracy. For example, a study comparing various classifiers found that Random Forest produced higher accuracy in phishing detection compared to Naive Bayes [19]

Naive Bayes serves as a foundational tool in phishing detection due to its simplicity and efficiency. While it may not always achieve the highest accuracy compared to more complex models, its ease of implementation and low computational requirements make it a valuable component in the arsenal against phishing attacks.

**Gated Recurrent Unit (GRU)**

The Gated Recurrent Unit (GRU), introduced by Cho et al. in 2014, is a gating mechanism in recurrent neural networks designed to capture dependencies of different time scales without having separate memory cells [20].

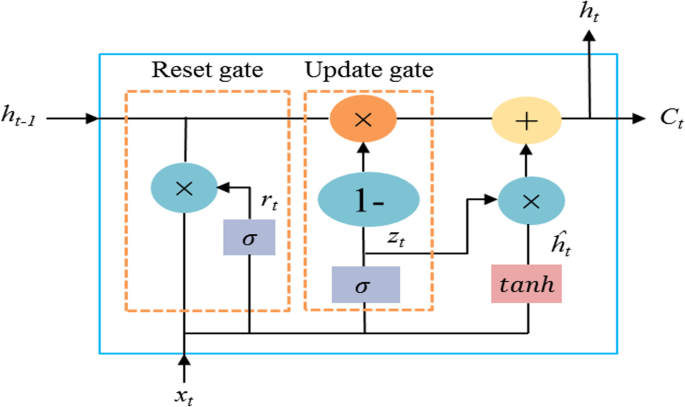


Figure 5: GRU Architecture

Update Gate: The update gate determines how much of the previous hidden state (information from the past) should be carried forward into the next state, and how much should be updated with new information from the current input.

It acts like a filter, balancing between retaining old information and incorporating new information.

Matthematically:



Reset Gate: The reset gate decides how much of the past information should be forgotten. This is especially useful when the model needs to ignore irrelevant past context for the current input.

By "resetting" certain parts of the hidden state, it enables the GRU to focus on relevant aspects of the input sequence.

Matthematically:



Candidate Activation: The candidate activation represents the potential new state that could replace the old state.

* It combines the reset gate's output with the current input and past state to compute a new candidate value.
* This ensures that the model adapts to new information while considering the context provided by the reset gate.
* Mathematically:

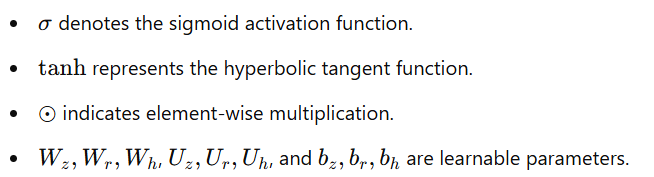


Hidden State: The hidden state is the final output of the GRU at the current step.

* It is a combination of the previous hidden state and the candidate activation, controlled by the update gate.
* The update gate decides how much of the candidate activation should replace the old state versus how much of the old state should be retained.
* Mathematically:



Where:



Applications in Phishing Detection

GRUs have been effectively applied in detecting phishing websites and URLs due to their capability to process sequential data and capture temporal dependencies.

A study proposed a hybrid approach using Long Short-Term Memory (LSTM) and GRU algorithms to detect phishing URLs, achieving an accuracy of 98.89%. [21]

Another research introduced a dual-layer Convolutional Neural Network (CNN) and GRU model with attention mechanisms for enhanced phishing website detection. [22]

Advantages of GRU in Phishing Detection

* Efficiency: GRUs are computationally less intensive due to fewer parameters, leading to faster training times.
* Capability to Capture Sequential Patterns: GRUs effectively model sequential dependencies, crucial for analyzing URLs and website content.
* Simplicity: The streamlined architecture of GRUs makes them easier to implement and tune compared to more complex models.

**Long Short-Term Memory (LSTM)**

Long Short-Term Memory (LSTM) networks were introduced by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradient problem in standard Recurrent Neural Networks (RNNs) [23]. LSTMs enhance RNNs by introducing memory cells and gating mechanisms, allowing them to retain long-term dependencies effectively. This ability makes LSTMs ideal for sequential data tasks, such as phishing detection in emails, URLs, and websites.

Key Idea: LSTMs are designed to selectively retain relevant information, discard unnecessary data, and dynamically update their memory over time. Their unique architecture, which includes forget, input, and output gates, allows them to handle long-term dependencies more effectively than traditional RNNs.

LSTM Architecture

LSTM cells consist of:

* Forget Gate: Determines what information from the previous time step should be forgotten.
* Input Gate: Decides which new information from the current input should be stored.
* Output Gate: Filters what part of the memory cell should be passed to the next state.
* Memory Cell: Maintains long-term dependencies by storing selective information over time.

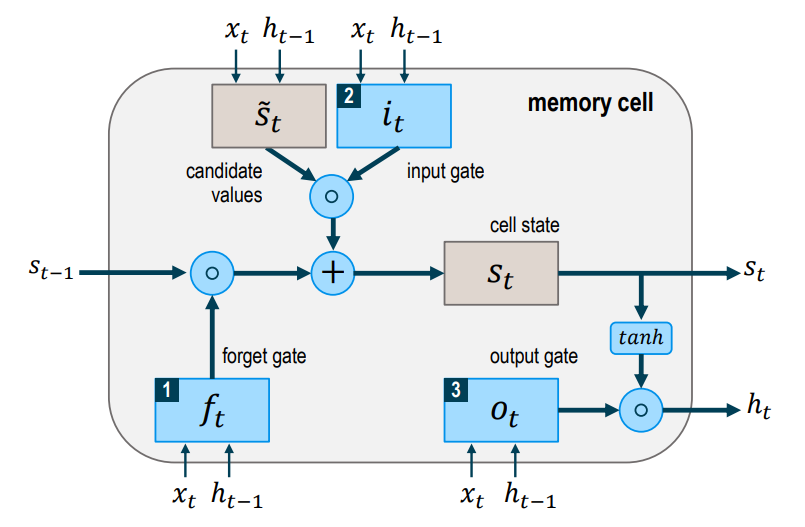


Figure 6: LSTM Architecture

These components collectively allow LSTMs to overcome the limitations of RNNs and excel in processing sequential data.

Applications of LSTM in Phishing Detection:

* Phishing Email Detection

LSTMs are used to analyze sequences of text in emails to identify phishing attempts.

By modeling long-term dependencies, they detect patterns such as suspicious keywords, impersonation, and unusual requests (e.g., "urgent action required").

Example: An LSTM network achieved 97.4% accuracy in phishing email detection by analyzing large email datasets for sequential patterns [24].

* Phishing URL Detection

LSTM networks are employed to process sequential data in URLs, identifying phishing links based on patterns like obfuscation, excessive subdomains, and hidden redirection.

Example: A study demonstrated that LSTM achieved 99.1% accuracy in phishing URL detection, outperforming other models [25].

* Phishing Website Detection

By analyzing the structure and metadata of websites, LSTM models can detect phishing sites, even those mimicking legitimate organizations.

Example: An LSTM-based detection system achieved 98.7% F1-score in classifying phishing websites by leveraging HTML and JavaScript features [26].

Advantages and Limitations

* Advantages: LSTM networks consistently outperform traditional machine learning models and even some advanced deep learning models in phishing detection.

Contextual Understanding: Can model complex sequential relationships, making them ideal for tasks like URL and email analysis.

Versatility: Applicable to various phishing scenarios (email, URLs, websites).

* Limitations: Requires significant training time and computational resources compared to simpler models like GRU.

Risk of Overfitting: Without sufficient regularization or diverse datasets, LSTM models may overfit smaller datasets.

Implementation Challenges: More complex architecture makes LSTM harder to implement and optimize than simpler models.

### Research Gap

Phishing email detection has significantly advanced with the integration of machine learning and deep learning techniques, yet critical gaps remain that hinder the development of a truly robust and efficient system. These gaps can be summarized as follows:

**Lack of Real-Time AI Integration**:  
While many AI-based phishing detection systems demonstrate high accuracy in offline evaluations, there is a significant lack of systems that effectively integrate artificial intelligence with real-time processing capabilities. Real-time phishing detection is crucial in practical scenarios to protect users from immediate threats, yet achieving this requires addressing challenges in latency, scalability, and efficient model inference.

**Comprehensive Comparison Between AI Models**:  
Current studies often focus on the implementation of specific models without providing a comprehensive comparison of their performance in similar conditions. For instance:

* + Naive Bayes models are rarely benchmarked alongside advanced deep learning models like LSTM, GRU, and Transformer-based approaches.
  + Detailed analysis of trade-offs, such as accuracy versus computational cost, is often missing. A holistic evaluation of these models is essential to guide the selection of the most suitable approach for specific use cases.

**Integration of Advanced AI Models**:  
While deep learning models such as LSTM and GRU are widely studied, cutting-edge models like GPT or Transformer-based architectures have not been fully explored in the context of phishing detection. Their ability to understand nuanced language patterns and contextual relationships offers significant potential for improving detection accuracy.

**Scalability and Cost Efficiency**:  
High-performing models like GPT achieve remarkable accuracy but incur substantial computational costs. This creates a need for scalable and cost-effective solutions that balance performance and efficiency, particularly for deployment in large-scale enterprise environments.

# Chapter 3: Methodology



## Proposed approach

This chapter outlines the design and implementation of the phishing email detection system, emphasizing the integration of multiple AI models—Naive Bayes, GRU, and LSTM—for detecting phishing attempts. Additionally, the system incorporates a comprehensive reporting mechanism to empower system administrators with insights and actionable data. This allows for effective email management and enhances user awareness through targeted training and awareness campaigns.

## System Architecture

The architecture integrates several core components that work in unison to deliver an end-to-end solution for phishing detection. At the heart of the system is the backend, which handles email processing, model inference, and database interactions. This backend communicates with the frontend, which provides a user-friendly interface for visualizing email classifications and generating reports. The database acts as a repository for storing processed emails, classification results, and audit logs, ensuring data persistence and accessibility.

The complete architecture is illustrated in Figure 7, highlighting the interaction between components such as the frontend, backend, database, and cloud services. This architecture not only facilitates robust phishing detection but also ensures the system is extensible and adaptable to evolving cybersecurity challenges.

Figure 7: Detail Architecture of the Web-Based Phishing Detection System

System Components:

Frontend:

* Provides a user-friendly interface for email analysis, classification results, and administrative dashboards.
* Includes visualization tools for reporting phishing trends and metrics.

Backend:

* Handles email processing, model inference, and API communication.
* Manages real-time classification and interaction with the database.

Database:

* Stores email metadata, classification results, and phishing reports for detailed analysis and auditing.
* Supports trend analysis over time.

Cloud Integration:

* Microsoft Azure enables scalable deployment of AI models and facilitates real-time email classification.
* Azure integrates with Microsoft Graph API for secure email handling.

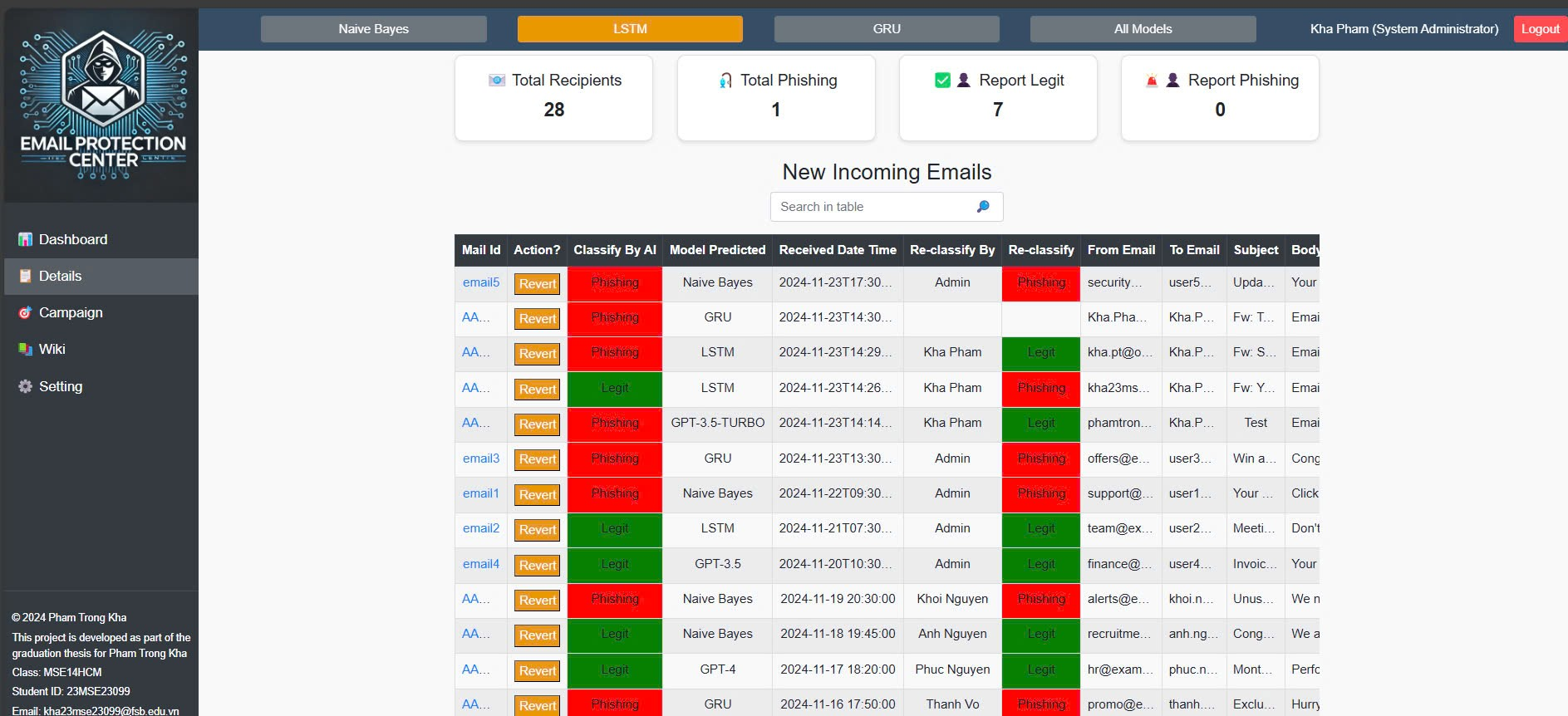


Figure 8: Web Application for Phishing Email Dectection

## Dataset

The dataset used in this study is a comprehensive compilation of phishing and legitimate emails sourced from multiple public repositories [5][27][]. To ensure that the dataset is suitable for training machine learning models and reflects real-world scenarios, several preprocessing and standardization steps were performed.

**Data Sources**

The dataset combines data from diverse and reliable sources, such as:

* Public cybersecurity repositories: Includes phishing email datasets published by academic and industry researchers.
* Enron Email Dataset: Provides legitimate corporate emails commonly used as a benchmark for email classification studies.
* Custom email collections: Additional data collected from simulated environments and anonymized email records.

**Data Structure**

The final dataset consists of two columns:

* Email Text: Contains the textual content of each email, including subject lines and body text. This serves as the primary input for the AI models.
* Email Type: Labels each email as either "Phishing Email" or "Safe Email" to indicate its classification.

**Dataset Balancing**

To create a balanced dataset that avoids bias during model training:

Equal representation of phishing and legitimate emails was ensured.

Figure 9 is the final dataset summary after balancing.

Total Emails: 14,250

Phishing Emails: 7,155 (50.2%)

Safe Emails: 7,095 (49.8%)

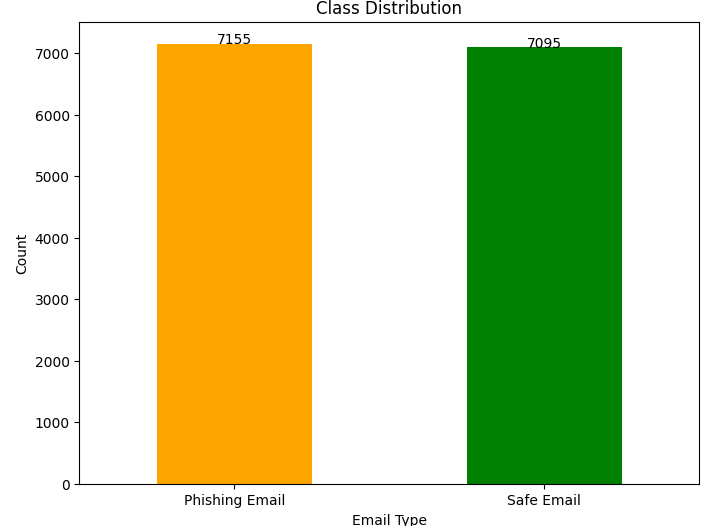


Figure 9: Dataset summary after balancing

This balanced distribution ensures that the models are not biased toward either class during training and evaluation. The Table 3 below represents for the sample of data in dataset.

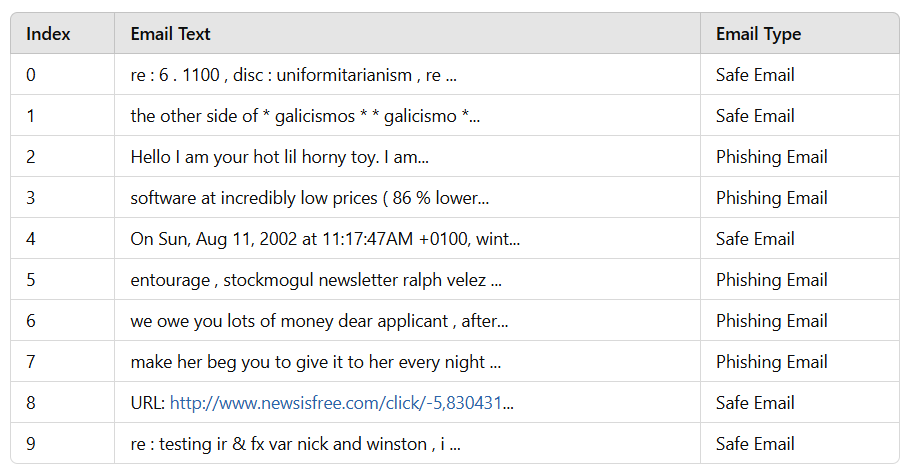


Table 3: Sample data in dataset.

**Analysis of Dataset Quality**

The dataset is well-structured with two columns: Email Text and Email Type. However, as shown in the word clouds in Figure 10, there are significant issues with the raw text data, including the presence of meaningless characters and special symbols that can affect the performance of the machine learning models if left unprocessed.

|  |  |
| --- | --- |
|  |  |

Figure 10: Wordcloud before preprocessing data

**Observations from Word Clouds:**

Safe Email Word Cloud:

The word cloud for safe emails contains several meaningful words such as "linux," "list," "info," and "thank," which are typical for professional or legitimate emails.

Phishing Email Word Cloud:

The word cloud for phishing emails highlights several suspicious keywords such as "money," "offer," "account," and "click," which align with typical phishing email patterns.

Data Quality Concerns: Special Characters and Artifacts: Both email types contain special characters and encoding artifacts (e.g., "Â," "Ã") that are irrelevant to the context of phishing detection. Words like "sourceforge" and "1/2" are not significant for classification and should be removed during preprocessing.

Effective preprocessing is a critical step to clean the data and extract meaningful features that enhance model performance.

**Data Preprocessing**

Data preprocessing is a crucial step in preparing raw text data for machine learning tasks, especially in the context of phishing email detection. Raw email data often contains noise, such as special characters, irrelevant symbols, and inconsistencies, which can negatively impact the performance of machine learning models. Therefore, preprocessing transforms unstructured and noisy data into a clean, structured format suitable for analysis.

According to Jurafsky and Martin [29], text preprocessing is foundational in natural language processing (NLP), as it ensures that the dataset is normalized, standardized, and free of unnecessary noise, thus enabling models to learn meaningful patterns effectively​. This preprocessing phase includes tasks such as tokenization, stopword removal, lemmatization, and feature extraction, which collectively improve the quality of the input data.

In this study, an extensive preprocessing pipeline was implemented, addressing common challenges in phishing datasets, such as the presence of HTML tags, URLs, email addresses, and encoding artifacts (e.g., "Â," "½"). Additionally, key phrases indicative of phishing attempts were highlighted to enhance feature relevance for classification models. These steps ensure that the dataset is both clean and representative of real-world phishing scenarios.

The preprocessing pipeline significantly reduces noise in the text data by standardizing and cleaning it.

The inclusion of phishing-specific phrases enhances the model's ability to detect patterns unique to phishing emails.

Generalization of placeholders (e.g., for dates and monetary values) helps prevent overfitting to specific instances in the dataset.

The detailed preprocessing steps are summarized in Table 4, which outlines each step and its corresponding role in data cleaning and preparation.



Table 4: Preprocessing Steps

**Preprocessing Results and Analysis**

Preprocessing transforms raw, unstructured email text into a cleaner and more structured format suitable for machine learning tasks. Table 5 illustrates the results of preprocessing applied to a sample of emails from the dataset. By addressing issues such as special characters, HTML tags, and irrelevant formatting, the pipeline ensures that the text retains its essential meaning while eliminating noise.

For example, URLs, email addresses, and other placeholders like DATE, MONEY, and EMAIL are standardized to prevent overfitting and provide consistency across the dataset. Redundant words such as stopwords are removed, and key phrases are highlighted to enhance the relevance of features extracted for phishing detection.

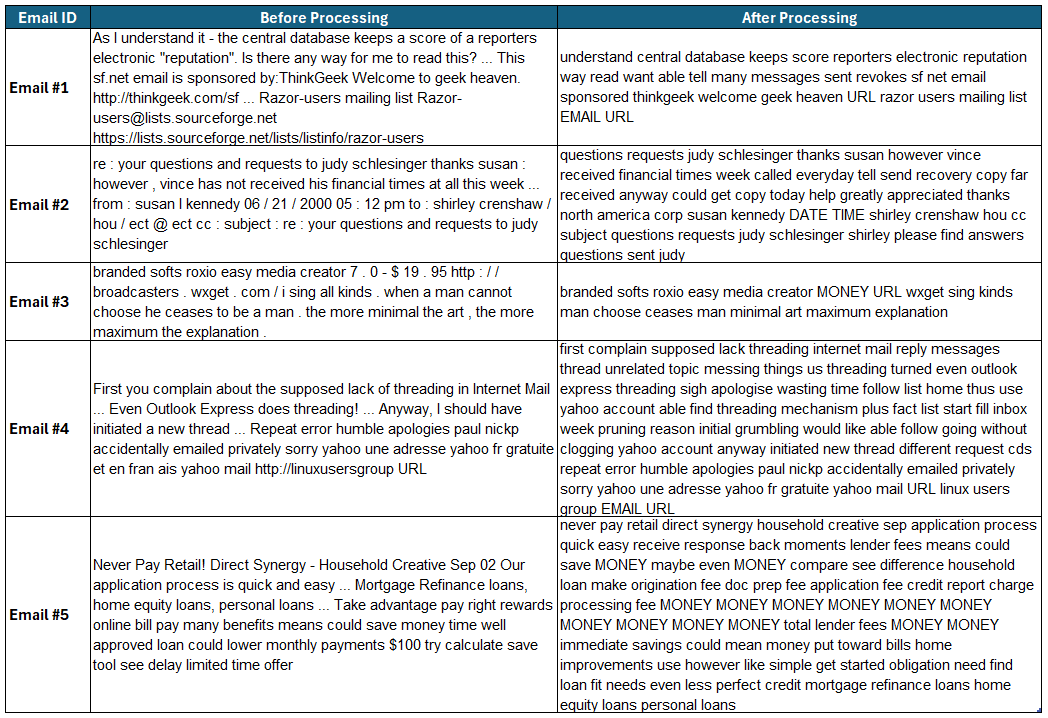


Table 5: Sample email after processing

Figure 11 presents word clouds generated from safe and phishing emails after preprocessing. These word clouds highlight the most frequently occurring tokens in the dataset, reflecting the impact of the preprocessing pipeline on the text. The results illustrate how preprocessing successfully reduces noise, standardizes data, and preserves meaningful patterns.

Preprocessing has successfully reduced noise while maintaining the semantic structure of the emails. Meaningful tokens dominate the word clouds, providing a clear distinction between safe and phishing emails.

The introduction of placeholders such as MONEY, URL, DATE, EMAIL, and PERCENTAGE ensures that the models focus on patterns rather than specifics, improving generalization and reducing overfitting.

Safe emails focus on professional or technical content, while phishing emails are centered around financial incentives and calls to action.

|  |  |
| --- | --- |
|  |  |

Figure 11: Wordcloud after preprocessing data

Safe emails after processing observation:

* Tokens like DATE, TIME, URL, and EMAIL dominate, representing common placeholders in legitimate communication.
* Words such as mail, system, list, and thank highlight professional and technical conversations focused on collaboration and issue resolution.
* Preprocessing removed irrelevant symbols (e.g., Â, ½) and introduced placeholders (DATE, URL, EMAIL) for consistent and meaningful patterns.

Phishing emails after processing observation:

* Tokens like MONEY, price, product, offer, and click emphasize financial deception and promotions.
* Words like address, free, and statement align with phishing attempts targeting personal or financial details.
* Preprocessing replaced monetary values and dates with placeholders (MONEY, DATE), consolidating redundant terms while preserving phishing-related patterns like financial exploitation (MONEY) and directives (click, offer).

Table 6 illustrates the top 20 most frequent words in safe emails before and after preprocessing, while Table 7 presents the corresponding results for phishing emails.

For safe emails in Table 6, the most frequent words before preprocessing include common stopwords like "the," "and," and symbols such as ".", "-", and ",". These contribute little value to phishing detection. After preprocessing, placeholders such as URL, EMAIL, and DATE dominate, alongside meaningful terms like list, information, and thanks, which reflect professional and collaborative communication.

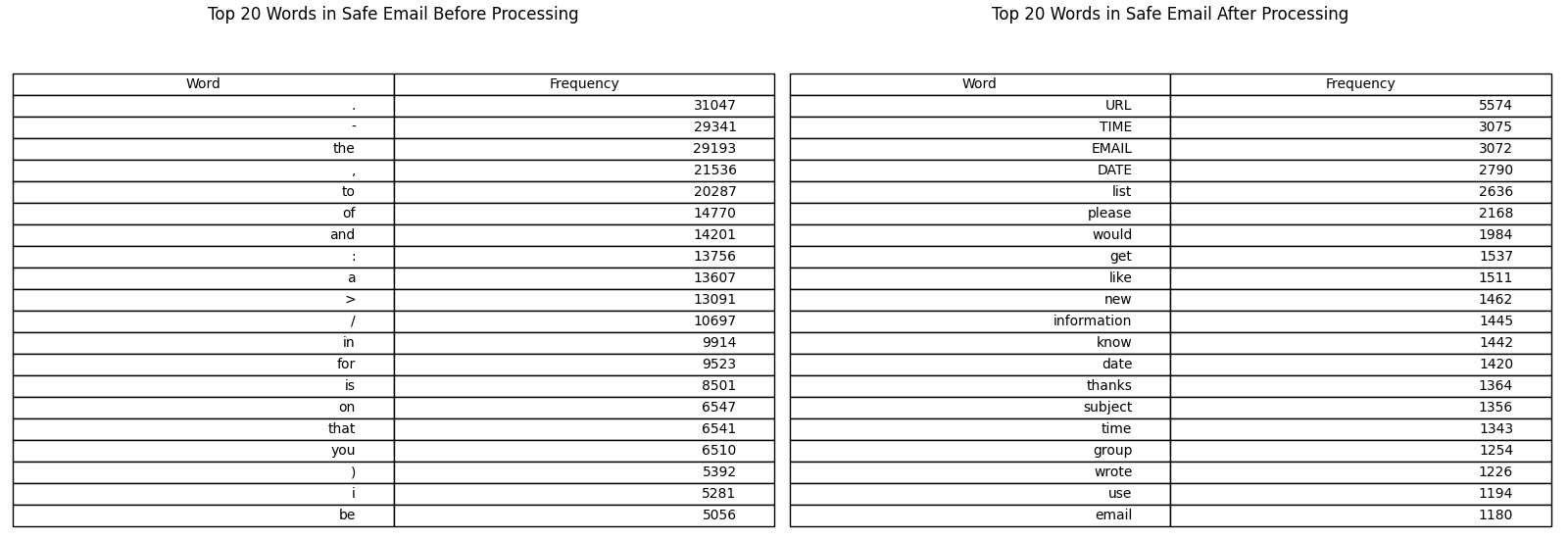


Table 6: Top 20 words in Safe Email before and after preprocessing

In Table 7, the top words in phishing emails show a similar pattern. Before preprocessing, stopwords and symbols dominate, masking the intent of phishing messages. After preprocessing, terms like MONEY, URL, free, and click become prominent, capturing the financial and deceptive nature of phishing emails.



Table 7: Top 20 words in Phishing Email before and after preprocessing

**Data Splitting**

Data splitting is a crucial step in machine learning workflows to ensure that the model is trained, validated, and tested on distinct subsets of the dataset. This process helps evaluate the model's performance on unseen data and prevents overfitting.

Training Set (70%): Used to train the model, allowing it to learn patterns and relationships in the data.

Validation Set (20%): Used to fine-tune model hyperparameters and evaluate intermediate performance during training.

Test Set (10%): Reserved for the final evaluation to assess how well the model generalizes to unseen data.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Figure 12: Data splitting into Training, Validation and Test sets.

## Implementation of Naive Bayes Model

The implementation employs a supervised machine learning approach using the Naive Bayes algorithm to detect phishing emails. Below are the key steps and methodologies involved, explained for thesis documentation purposes.

**Text Vectorization**

The text data is converted into numerical features using TF-IDF Vectorization. This technique assigns importance to words based on their frequency in a document relative to their frequency across the entire dataset.

Steps:

* Max Features: The TF-IDF vectorizer is restricted to a maximum of 5000 features, ensuring computational efficiency and avoiding sparsity.
* Figure 13 show the top 20 words with the highest TF-IDF scores are identified and visualized using a bar plot. These keywords indicate the most distinguishing terms in the dataset.

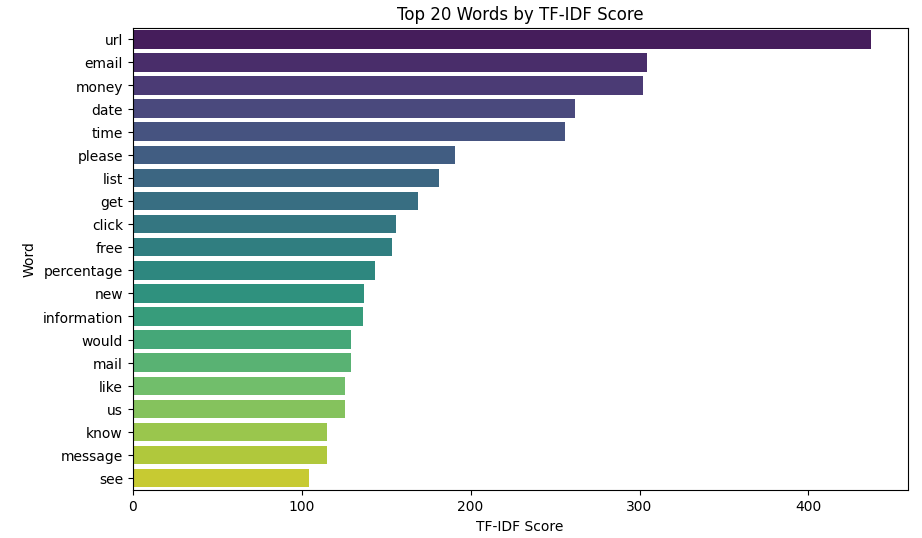


Figure 13: Top 20 Words by TF-IDF Score

**Model Training and Evaluation**

A Multinomial Naive Bayes model is trained on the vectorized training data.

Evaluation Metrics:

Confusion Matrix: A confusion matrix visually compares actual and predicted labels, providing insights into true positives, false positives, and other error categories.

Classification Report: Precision, recall, F1-score, and support for each class (Safe, Phishing) are calculated to understand the model's performance.

Accuracy: The overall accuracy of the model is measured, indicating the proportion of correctly classified emails.

|  |  |
| --- | --- |
|  |  |

Figure 14: Confusion Matrix and Classification report

**Evaluation of Metrics from Figure 14**

Confusion Matrix

The confusion matrix provides the following performance metrics:

* True Positives (TP): 1390 emails correctly classified as phishing.
* True Negatives (TN): 1353 emails correctly classified as safe.
* False Positives (FP): 66 emails mistakenly classified as phishing.
* False Negatives (FN): 41 emails mistakenly classified as safe.

Analysis:

* High True Positive and True Negative Rates: The majority of predictions are correct, with a total of 2743 out of 2850 samples classified accurately.
* Low False Positives (FP): Only 66 emails were falsely flagged as phishing, reducing unnecessary alerts.
* Low False Negatives (FN): Only 41 phishing emails were missed, demonstrating the model's strong recall for phishing detection.

Classification Report

The classification report breaks down performance metrics per class (Safe and Phishing):

Class 0 (Safe):

* Precision: 0.97, meaning 97% of emails classified as Safe were correct.
* Recall: 0.95, indicating 95% of actual Safe emails were identified.
* F1-Score: 0.96, a harmonic mean of precision and recall.

Class 1 (Phishing):

* Precision: 0.95, meaning 95% of emails classified as Phishing were correct.
* Recall: 0.97, indicating 97% of actual phishing emails were identified.
* F1-Score: 0.96, showing balanced performance.

Overall Metrics:

* Accuracy: 96.25%, reflecting that over 96% of all emails were classified correctly.
* Macro Average (Precision, Recall, F1-Score): 0.96, showing that both classes perform equally well on average.
* Weighted Average: 0.96, accounting for the class distribution in the dataset.

**Summary**

* The metrics indicate excellent classification performance:
* Precision and recall are consistently high across both classes, demonstrating balanced and effective classification.
* The low False Positive and False Negative rates ensure reliable detection with minimal errors.
* An overall accuracy of 96.25% supports the model's effectiveness for distinguishing between safe and phishing emails.

The trained Naive Bayes model and the TF-IDF vectorizer have been successfully saved, ensuring the reproducibility and deployment readiness of the system. These saved components will be utilized for evaluating the model's performance on real-world data in the subsequent chapter, where its practical applicability and effectiveness will be assessed in a realistic scenario.

## Implementation of GRU Model

This section describes the implementation of a Gated Recurrent Unit (GRU) model for phishing email detection. The GRU model leverages deep learning to process sequential text data and classify emails as either "Safe" or "Phishing." Below are the key steps in the workflow, along with the rationale for selecting the parameters.

**Data Preparation**

* Tokenization and Padding

Text data must be converted into numerical representations for the GRU model:

* + Tokenization: The text data is tokenized using the Tokenizer to convert each word into an integer ID, preserving word order.
  + Padding: To ensure consistent input size for the GRU, sequences are padded to a fixed length using pad\_sequences. Shorter sequences are padded with zeros, while longer ones are truncated.
* Text Length Analysis:
  + Maximum Sequence Length (max\_length=256): Selected based on text length analysis, which shows most emails fall below this length. This ensures the model captures sufficient text information without excessive computational cost.
  + The average text length is 112 words, while the maximum length is 6325 words.
  + The 75th percentile of text lengths is 102 words, indicating most emails are relatively short. The chosen max\_length=256 effectively accommodates these variations.

|  |  |
| --- | --- |
| count | 9975.000000 |
| mean | 112.391278 |
| std | 233.604657 |
| Min | 0.000000 |
| 25% | 32.000000 |
| 50% | 59.000000 |
| 75% | 102.000000 |
| Max | 6325.000000 |

Table 6: Statistical Summary of Email Text Lengths

**GRU Model Architecture**

The GRU model is designed to efficiently process sequential text data. After experimenting with different parameter combinations, the following configuration was chosen for its performance and efficiency:

Embedding Layer

* Purpose: Converts tokenized sequences into dense vector representations that capture semantic relationships between words.
* Parameter:

Embedding Dimension (embed\_size=200): A dimension that balances detail and computational efficiency. Experimentation showed that higher dimensions (e.g., 300) did not significantly improve results, while lower dimensions (e.g., 100) reduced accuracy.

GRU Layers

* Purpose: GRU layers process sequential data and learn temporal dependencies, which are essential for text classification.
* Parameters:

Hidden Units (hidden\_size=256): Provides sufficient capacity for learning complex relationships in text. Smaller sizes (e.g., 128) led to reduced accuracy, while larger sizes increased training time with diminishing returns.

Number of Layers (n\_layers=2): Stacking two GRU layers allows the model to learn hierarchical features. Additional layers (e.g., 3 or more) increased computation without significant performance gains.

Dense and Dropout Layers

* Dense layers improve representation learning, while dropout layers reduce overfitting.
* Configuration:

Dropout (40%): Applied after the GRU layers to enhance generalization.

Dense (128 nodes): Captures complex, high-dimensional patterns in text.

Adding more nodes (e.g., 256) showed no significant improvement.

Dense (64 nodes): A smaller dense layer consolidates learned features.

Dropout (20%): Applied to the dense layers to further mitigate overfitting.

Output Layer

* Purpose: Provides binary classification output for "Safe" and "Phishing" labels.
* Configuration:

1 Node: Represents the probability of an email being phishing.

Sigmoid Activation: Outputs a value between 0 and 1 for binary classification.

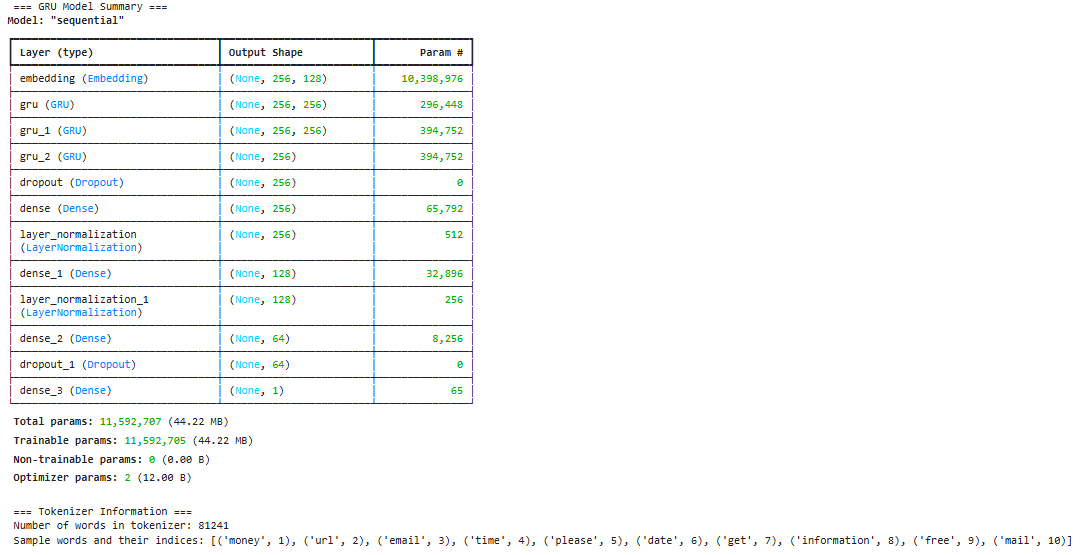


Figure 15: GRU Model Summary

**Training Process**

The model is compiled with:

* Loss Function:

Binary Crossentropy: Suitable for binary classification tasks, measuring the difference between true and predicted probabilities.

* Optimizer:

Adam: Selected for its adaptive learning rate, which speeds up convergence while maintaining stability.

* Learning Rate (learning\_rate=1e-4): A small learning rate ensures stable and gradual convergence. Larger rates (e.g., 1e-3) led to unstable training, while smaller rates (e.g., 1e-5) significantly increased training time.

To optimize training and prevent overfitting, the following callbacks are used:

* Early Stopping:

Monitors validation loss and halts training if it does not improve for 5 consecutive epochs.

Restores the best weights observed during training.

* ReduceLROnPlateau:

Dynamically reduces the learning rate by half if the validation loss stagnates for 2 epochs.

Ensures finer updates to weights during later epochs.

Batch Size and Epochs

* Batch Size (batch\_size=64): Balances memory efficiency and gradient estimation accuracy. Smaller batches (e.g., 32) required more iterations per epoch, while larger batches (e.g., 128) reduced convergence speed.
* Epochs (epochs=100): Allows sufficient iterations for the model to converge. Early stopping ensures that training halts once validation loss plateaus.

**Visualization of Training Metrics**

During training, both loss and accuracy are tracked and visualized:

|  |  |
| --- | --- |
|  |  |

Figure 16: GRU Loss and Accuracy over epochs

Loss Curves:

* Training loss decreases sharply in the early epochs and converges to near-zero.
* Validation loss stabilizes after epoch 3, indicating the model generalizes well.

Accuracy Curves:

* Training accuracy rapidly increases to near-perfect levels.
* Validation accuracy stabilizes around 97%, showing strong generalization to unseen data.

**Evaluation**

The high test accuracy and low test loss confirm the model's effectiveness in distinguishing phishing emails from safe emails. The minimal difference between training, validation, and test metrics suggests the model generalizes well to new data.

|  |  |
| --- | --- |
|  |  |

Figure 17: GRU Confusion Matrix and Classification Report

The confusion matrix and classification report demonstrate strong performance of the GRU model on the validation dataset:

Confusion Matrix:

* True Positives (TP): 1386 phishing emails correctly classified.
* True Negatives (TN): 1342 safe emails correctly classified.
* False Positives (FP): 77 safe emails misclassified as phishing.
* False Negatives (FN): 45 phishing emails misclassified as safe.

Classification Report:

* Precision:

Safe: 97%, Phishing: 95% (Overall: Excellent precision for both classes).

* Recall:

Safe: 95%, Phishing: 97% (Indicates the model captures phishing emails effectively).

* F1-Score:

Both classes achieve 96%, showing a balance between precision and recall.

* Accuracy: 95.72%, reflecting the overall correct classification rate.

Insights:

* The model performs exceptionally well, achieving a high accuracy of 95.72% and balanced precision and recall across both classes.
* Low false negatives (45) indicate the model's ability to detect phishing emails effectively, which is critical for minimizing security risks.

**Model and Tokenizer Saving**

* To ensure reusability, the trained GRU model and tokenizer are saved:
* GRU Model: Serialized as an .h5 file for deployment.
* Tokenizer: Saved as a .pkl file to preprocess new text data consistently.

## Implementation of LSTM Model

The LSTM (Long Short-Term Memory) model implementation closely resembles the GRU-based implementation, but it incorporates key differences that address sequential dependencies and provide enhanced control over long-term memory. Below, we highlight the differences and enhancements in the LSTM implementation, focusing on its structure, parameter adjustments, and performance results.

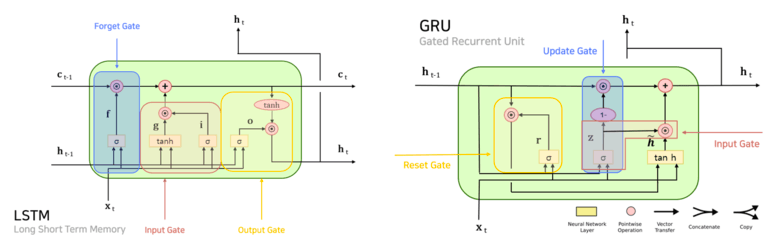


Figure 17: GRU and LSTM architectures

Key Differences Between LSTM and GRU Implementation

LSTM Architecture

* Cell Structure: Unlike GRU, LSTM has additional gating mechanisms (input, forget, and output gates) that explicitly control the flow of information through time steps. This makes LSTM particularly effective for longer sequences or datasets with complex temporal dependencies.
* Configuration:

Number of Layers (n\_layers=3): The LSTM model uses three stacked LSTM layers compared to two GRU layers, allowing for deeper hierarchical feature extraction.

Dropout: Regularization is applied after each LSTM layer with a dropout rate of 30% to prevent overfitting.

Additional Dense Layers

* Purpose: Enhance the model's ability to capture non-linear relationships and further refine features.
* Changes: Two dense layers with 128 units (ReLU activation) and additional dropout (40%) were added.

Hyperparameter Adjustments

* Embedding Dimension (embed\_size=200): The higher embedding size accommodates the complexity of the LSTM model and captures richer word relationships.
* Batch Size (batch\_size=64): Reduced from GRU's configuration to manage computational load given LSTM's increased complexity.

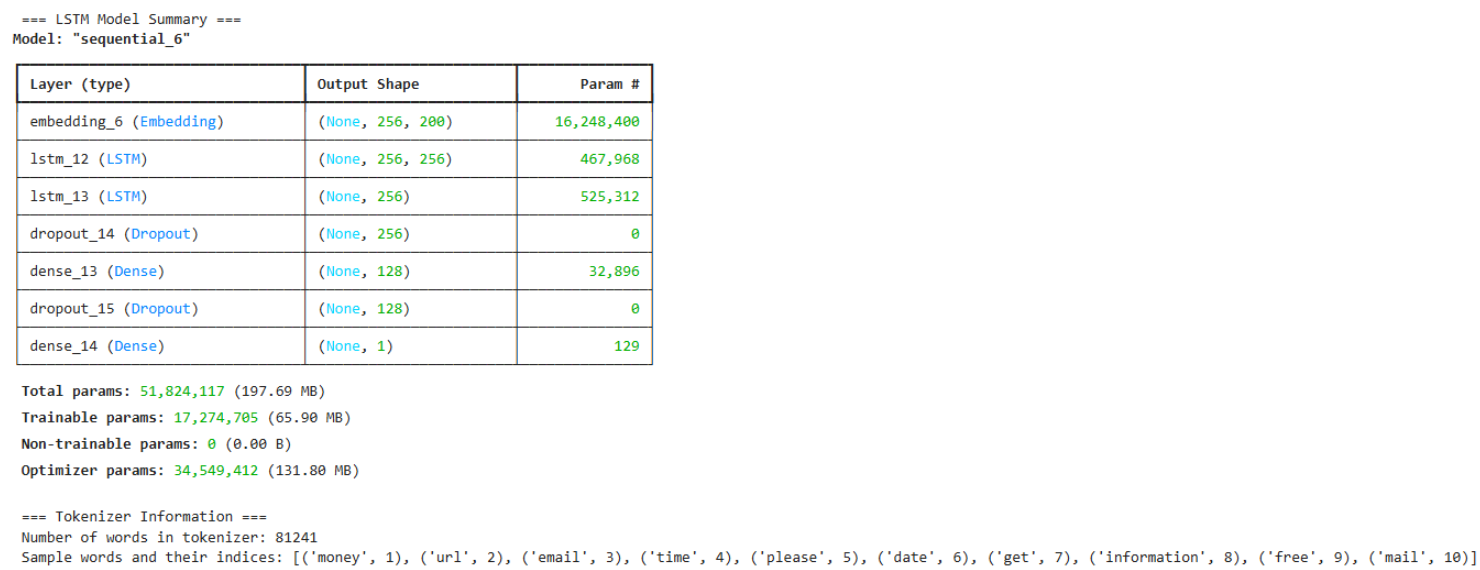


Figure 18: LSTM Model Summary

**Visualization of Training Metrics**

During training, both loss and accuracy are tracked and visualized:

|  |  |
| --- | --- |
|  |  |

Figure 19: LSTM Loss and Accuracy over epochs

**Training Results**

Training and Validation Metrics:

* Training Accuracy: Rapidly increases, achieving near-perfect accuracy (99.7%)
* Validation Accuracy: Stabilizes at 96.8%, indicating strong generalization to unseen data.
* Training Loss: Decreases consistently, reaching near-zero, while validation loss stabilizes after initial improvement.

Learning Rate Adjustment:

* The ReduceLROnPlateau callback reduces the learning rate after epoch 6 when validation loss plateaus, enabling finer updates for convergence.

**Evaluation Results**

|  |  |
| --- | --- |
|  |  |

Figure 20: LSTM Confusion Matrix and Classification report

Confusion Matrix and Classification Report (Validation Set):

* True Positives (TP): 1380 phishing emails correctly classified.
* True Negatives (TN): 1381 safe emails correctly classified.
* False Positives (FP): 38 safe emails misclassified as phishing.
* False Negatives (FN): 51 phishing emails misclassified as safe.
* Accuracy: 96.88%, reflecting excellent overall performance.
* Precision, Recall, and F1-Score: Consistently high for both classes (Safe and Phishing), each achieving around 97%.

Test Set Evaluation:

* Accuracy: 96.84%
* Test Loss: 0.131
* These metrics confirm that the LSTM model performs reliably on unseen data, comparable to its performance on validation data.

**Performance Comparison: LSTM vs. GRU**

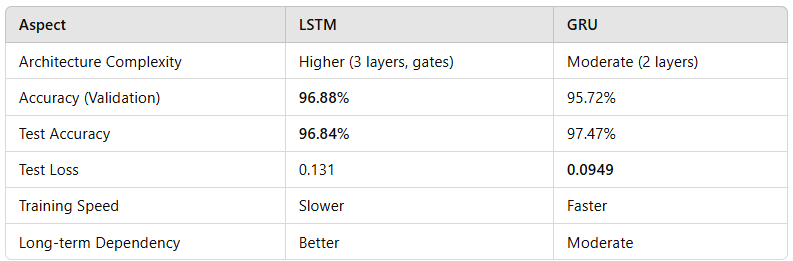


Table 7: Performance Comparison: LSTM vs. GRU

**Conclusion**

While both models achieved excellent results, the LSTM model provides slightly better control over long-term dependencies, making it more suitable for datasets with complex sequential relationships. However, the GRU model offers comparable performance with faster training time and fewer computational resources.

The final choice should be based on real-world testing with phishing emails and unseen data to evaluate their generalization, robustness, and efficiency in practical applications.

## Frontend Implementation for Phishing Detection System

This section describes the implementation of the frontend for the phishing detection system. The frontend serves as the primary interface for users to interact with the system, visualize phishing email statistics, and manage data. The design prioritizes usability, responsiveness, and seamless integration with backend APIs.

**Overview of the Frontend Design**

The frontend is developed using HTML, CSS, and JavaScript, with additional support from Bootstrap for responsive layouts. It comprises three main pages:

* Login Page: Provides authentication for users to access the system.
* Dashboard Page: Displays key statistics, interactive charts, and model management options.
* Details Page: Offers detailed information about emails, including classification and reclassification features.

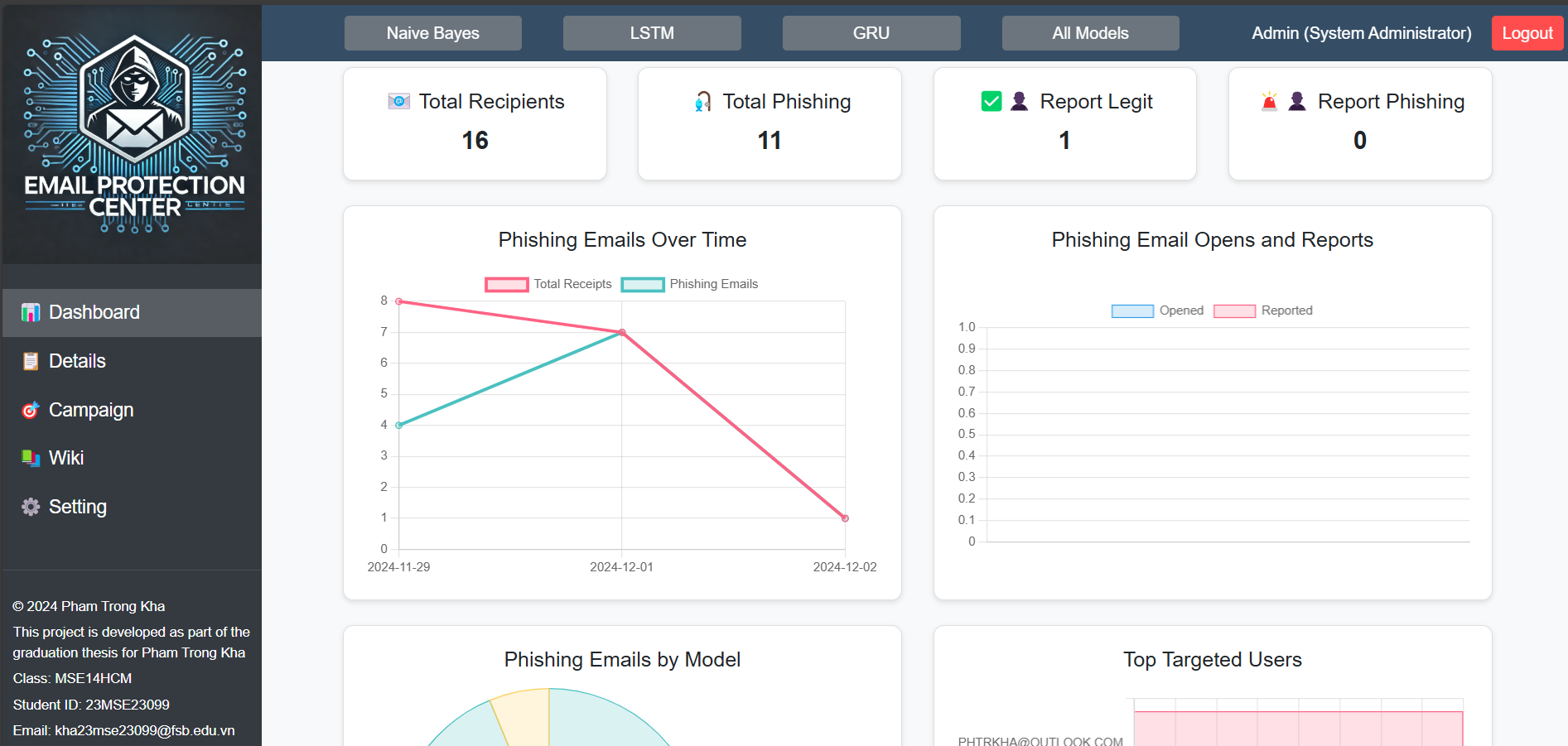


Figure 21: Email Protection Center Dashboard Page

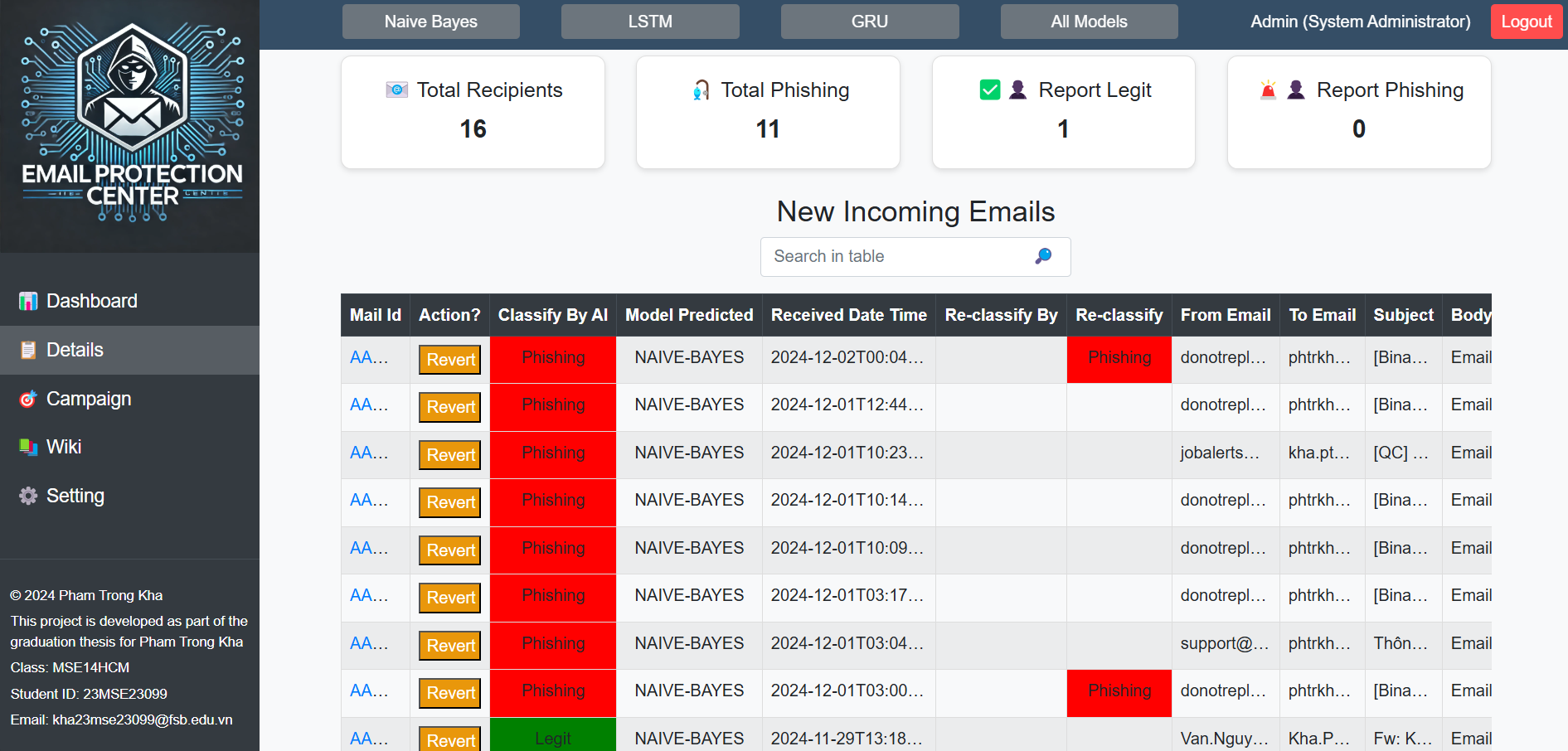


Figure 22: Email Protection Center Details Page

These components are styled using a custom CSS file and JavaScript to enable dynamic functionality and API integration.

**Key Features and Functionality**

**Login Page**: The login page is the entry point to the system. It ensures that only authenticated users can access the dashboard and related features.

* Functionality:

Verifies credentials against a pre-defined list of users in JavaScript.

Stores session data (username, role, and fullName) in localStorage.

Redirects authenticated users to the dashboard.

Displays error messages for invalid credentials.

* Design:

Simple, clean layout with a logo, input fields, and a login button.

Inline CSS ensures quick rendering and easy adjustments.

**Dashboard Page**: The dashboard is the primary interface where users can view system statistics and manage machine learning models.

Core Features:

* Model Management:

Allows users to select and activate models (Naive Bayes, LSTM, GRU, or All Models).

Updates the active model via API calls and reflects changes visually.

* Key Metrics:

Displays total recipients, phishing and user-reported phishing cases.

Data is dynamically fetched and updated in real-time.

* Interactive Charts:

Visualizes phishing trends, user interaction, and model performance using various charts.

Data is refreshed at regular intervals to ensure accuracy.

* Navigation:

Sidebar links to other pages (Details, Campaigns, Wiki, Settings).

Navbar includes user information and logout functionality.

**Details Page:** The details page provides comprehensive information about individual emails and allows user interaction for reclassification.

Core Features:

* Email Table:

Displays attributes such as sender, recipient, subject, and classification.

Allows users to filter emails using a search bar.

* Email Reclassification:

Users can manually reclassify emails by toggling their classification status.

Sends updated data to the backend for storage and further analysis.

* Email Popup:

Displays detailed content of an email in a popup window for easy review.

**Dynamic Behavior and API Integration**

The frontend interacts with the backend using RESTful APIs to fetch and update data dynamically. JavaScript handles the following:

* Session Management:

Checks for active sessions using localStorage.

Redirects unauthenticated users to the login page.

* Data Fetching:

Fetches statistics (e.g., phishing trends, email counts) and updates charts in real-time.

Retrieves detailed email data for display in the details page.

* User Actions:

Sends API requests for model selection and email reclassification.

Updates the UI dynamically based on backend responses.

* Error Handling:

Logs errors in the console and provides minimal disruption to user experience.

**Styling and Responsiveness**

Custom styles defined in the style.css file to ensure a consistent and visually appealing layout:

* Sidebar and Navbar:

Fixed positioning for intuitive navigation.

Distinct hover and active states for links.

* Cards and Charts:

Shadows and rounded corners for a modern look.

Flexbox layout ensures compatibility across devices.

* Tables:

Responsive design with scrollable containers for large datasets.

Ellipsis styling for truncated content.

**Integration with Backend**

The frontend communicates with the backend through the following API endpoints:

* Model Management:

Activates models via /api/model/setup and retrieves the current active model from /api/model/current.

* Data Fetching:

Retrieves email statistics, classifications, and detailed email information.

* Email Reclassification:

Updates email classification using /api/classify\_email.

**Conclusion**

The frontend of the phishing detection system is intuitive, responsive, and integrates seamlessly with the backend, supporting real-time updates, model management, and email reclassification. Future enhancements could include secure authentication and optimized data fetching for scalability.

## Azure Graph API and Webhook Intergration

This section focuses on the integration of Azure Graph API and webhook services into the phishing detection system. These components facilitate seamless data acquisition and real-time updates, enabling the system to process emails dynamically and respond promptly to new data.

**Azure Graph API Integration**

Overview

Azure Graph API is a powerful tool for accessing data within Microsoft 365 services, such as Outlook. It provides endpoints for fetching emails, users, and other organizational resources, which are essential for detecting phishing emails.

Use Case in Phishing Detection

* Fetch emails from Microsoft 365 mailboxes for analysis.
* Access metadata, such as sender, recipient, subject, and email content, for phishing detection.
* Retrieve user information for personalized reports and targeted notifications.

Implementation

* Authentication:

Uses OAuth 2.0 to obtain access tokens.

Tokens are used to authorize requests to the Graph API.

* Fetching Emails:

API endpoint: /me/messages or /users/{id}/messages.

Includes query parameters for filtering (e.g., unread emails, specific folders).

Example: GET https://graph.microsoft.com/v1.0/me/messages?$filter=isRead eq false

Processing Data:

* Extract metadata (e.g., sender, subject, content).
* Convert raw data into a format suitable for model input.

Security Considerations

* Token Expiry: Refresh tokens periodically to maintain access.
* Permission Scopes: Minimize permissions (e.g., Mail.Read) to reduce security risks.
* Data Encryption: Ensure data is encrypted both in transit and at rest.

**Webhook Integration**

Overview

Webhooks allow the system to receive real-time notifications about specific events, such as the arrival of new emails. Instead of polling the API, webhooks push updates, improving efficiency.

Use Case in Phishing Detection

* Notify the system when new emails arrive in a user's inbox.
* Trigger immediate processing of emails for phishing detection.
* Update dashboards and notify users in real-time.

Implementation

Subscription Setup:

* Create a subscription to monitor changes in the mailbox.
* Example request to subscribe to new emails:

POST https://graph.microsoft.com/v1.0/subscriptions

{

"changeType": "created",

"notificationUrl": "https://yourdomain.com/webhook",

"resource": "/me/messages",

"expirationDateTime": "2024-01-01T00:00:00Z"

}

Webhook Endpoint:

* A server endpoint listens for notifications from Azure.
* Processes incoming notifications and retrieves detailed email data via the Graph API.

Notification Handling:

* Parse incoming data to identify relevant emails.
* Example webhook payload:

{

"value": [

{

"resource": "/users/{id}/messages/{message-id}",

"changeType": "created",

"clientState": "secret-value"

}

]

}

Error Handling:

* Respond with a 200 OK to acknowledge receipt.
* Log and retry failed webhook notifications.

Security Considerations

* Validation: Verify the clientState to ensure the notification originates from Azure.
* HTTPS: Secure the webhook endpoint with HTTPS.
* Access Control: Restrict access to the endpoint to authorized IPs.

**Integration Workflow**

* **Data Flow**:

Azure Graph API fetches emails and user details.

Webhooks trigger real-time updates when new emails arrive.

* **Backend Coordination**:

Webhook notifications initiate email analysis.

Results are stored in the database and displayed in the frontend.

* **Real-Time Updates**:

Dashboards and notifications are updated dynamically based on webhook triggers.

**Conclusion**

The integration of Azure Graph API and webhooks enhances the phishing detection system by automating data retrieval and enabling real-time updates. This approach reduces latency, improves efficiency, and ensures timely detection of phishing emails.

## Backend Implementation

In the previous sections, we discussed building machine learning models, developing the frontend, and configuring the Azure Graph API. Now, we will dive into the backend implementation to see how all these components interact seamlessly.

The backend acts as the central coordinator, integrating the frontend, database, machine learning models, and Azure Graph API while ensuring smooth communication between them.

**Overview of Backend Architecture**

The backend is designed using Flask, a Python web framework, and is supported by several modules to handle real-time email classification, data management, and API communication. It integrates machine learning models for phishing detection, interacts with the MySQL database, and processes Azure Graph API data, ensuring seamless operation of the entire system.

The backend architecture, as illustrated in **Figure 7**, represents the core design and workflow of the phishing detection system.

Figure 7: Detail Architecture of the Web-Based Phishing Detection System

**Backend Workflow**

Frontend Interaction:

* The frontend sends requests for email statistics, phishing trends, and model selection.
* Backend APIs process these requests and return the required data.

Email Classification:

* Emails are fetched from Azure Graph API or processed through webhook notifications.
* The active machine learning model classifies the emails, and results are stored in the database.

Real-Time Data Handling:

* Webhook notifications ensure that new emails are processed and classified immediately.
* The frontend retrieves updated statistics and classifications for user review.

**Key Features of the Backend**

**1. Frontend Interaction**

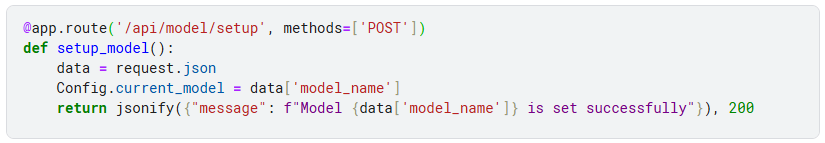
The backend exposes RESTful APIs to communicate with the frontend:

Model Management:

* Users can dynamically switch between models (GRU, LSTM, Naive Bayes).
* Relevant file: app.py

Endpoint: /api/model/setup

This endpoint allows the frontend to update the active model dynamically



* Endpoint: /api/model/current

Returns the currently active model for frontend display.

Email Data:

* The backend provides email metadata, classification results, and phishing trends to the frontend for real-time visualization.
* Relevant file: email\_logs\_manager\_v2.py
  + Function: get\_phishing\_trends()
    - Fetches trends based on the number of phishing emails detected over time.

**2. Machine Learning Models**

The backend integrates GRU, LSTM, and Naive Bayes models, supporting dynamic model selection for email classification:

* Dynamic Model Execution:

The currently active model processes email content to classify emails as phishing or safe.

* + Relevant files: gru.py, lstm.py, naive\_bayes.py

Each file contains a predict\_email function for classification:



* Model Switching:

Users can choose a specific model via the frontend, and the backend adjusts its configuration accordingly.

**3. Azure Graph API Integration**

Azure Graph API enables the backend to fetch email data from Microsoft 365 accounts:

* Email Fetching:
  + Retrieves unread emails from user inboxes for processing.
  + Relevant file: app.py
  + Function: fetch\_user\_emails()

Communicates with Azure Graph API using OAuth 2.0 tokens:



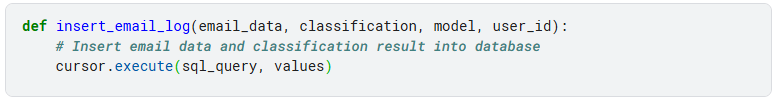
* Webhook Subscriptions:
* Registers webhook subscriptions to receive notifications about new or updated emails.
* Relevant file: app.py
  + Function: register\_mail\_webhook\_for\_user()

Handles webhook subscriptions for specific events (e.g., new email arrivals).

**4. Database**

The backend interacts with a MySQL database to store and manage email logs:

* Logging:
  + Classification results and metadata are logged in the database for future reference.
  + Relevant file: email\_logs\_manager\_v2.py
  + Function: insert\_email\_log():



* Data Retrieval:
  + Fetches email statistics and trends for display on the frontend.
  + Function: get\_statistics()
  + Retrieves the total count of phishing emails, safe emails, and user-reported cases.

**5. Webhook Notifications**

Webhooks enable real-time email updates:

* Notification Handling:
  + Listens for webhook events triggered by Azure when new emails arrive.
  + Relevant file: app.py

Endpoint: /webhook

Processes incoming notifications and fetches email details for classification:



* Real-Time Updates:
  + Upon classification, results are stored in the database and updated on the frontend.

**Conclusion**

The backend architecture effectively integrates machine learning models, Azure Graph API, and a MySQL database, providing robust functionality for phishing detection.

By coordinating all components, the backend ensures real-time data processing, dynamic model management, and seamless communication with the frontend.

Relevant modules such as app.py, email\_logs\_manager\_v2.py, and the machine learning scripts (e.g., gru.py, lstm.py) exemplify how the backend achieves these functionalities. This design ensures scalability and adaptability for future system enhancements.

# Chapter 4: Results and Discussion



## Results

In this section, the phishing detection system is evaluated through a two-step process. First, the machine learning models (GRU, LSTM, and Naive Bayes) are tested on a subset of 10 randomly selected emails from the validation dataset to verify their classification performance.

Second, the models are tested with real-world email data fetched dynamically using Azure Graph API. These tests demonstrate the system's end-to-end functionality, from model predictions to real-time integration with user workflows.

### Model Evaluation

To verify the accuracy and behavior of each machine learning model, 10 emails were randomly selected from the validation dataset. These emails include a mix of legitimate and phishing samples to test the models' ability to handle diverse content.

* **Email Selection**:

10 emails were chosen randomly from the validation dataset.

These emails include varied characteristics, such as promotional content, transactional emails, and known phishing templates.

* **Model Execution**:

Each email was processed using the GRU, LSTM, and Naive Bayes models.

Predictions included the classification label ("Phishing" or "Safe") and confidence scores.

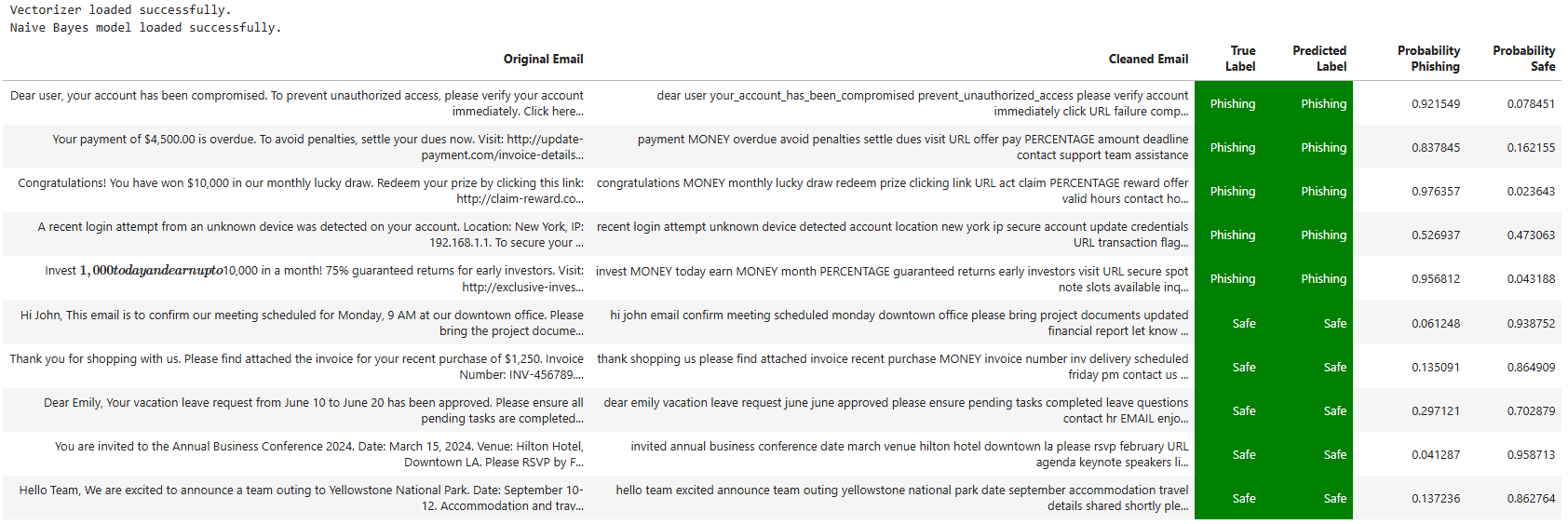


Table 8: Naïve Bayes Model Classification

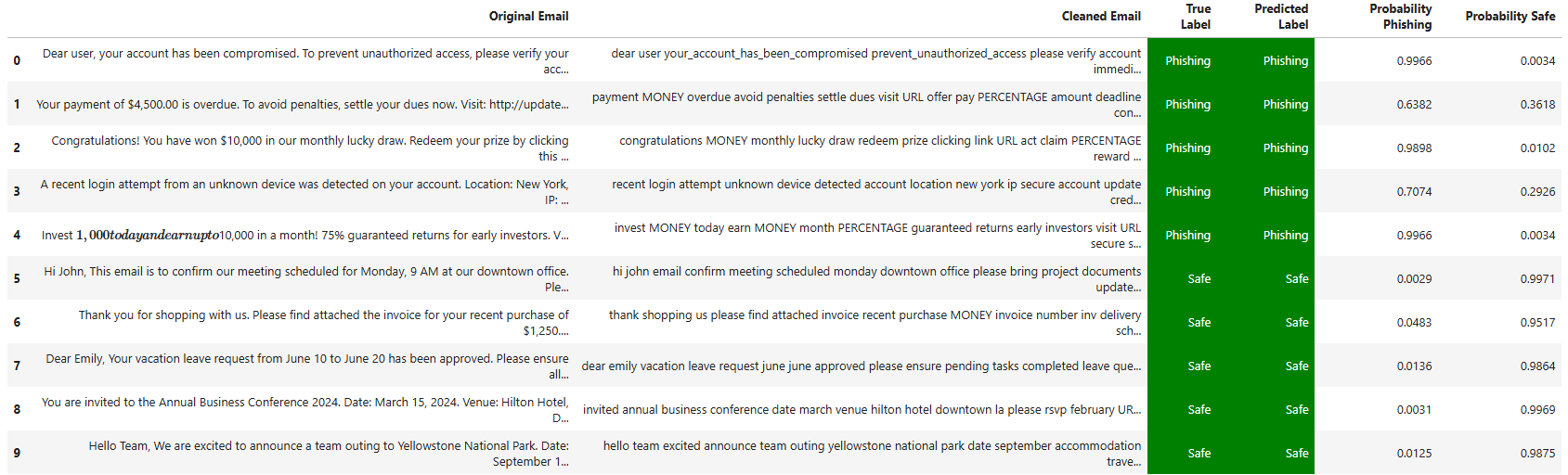


Table 9: GRU Model Classification

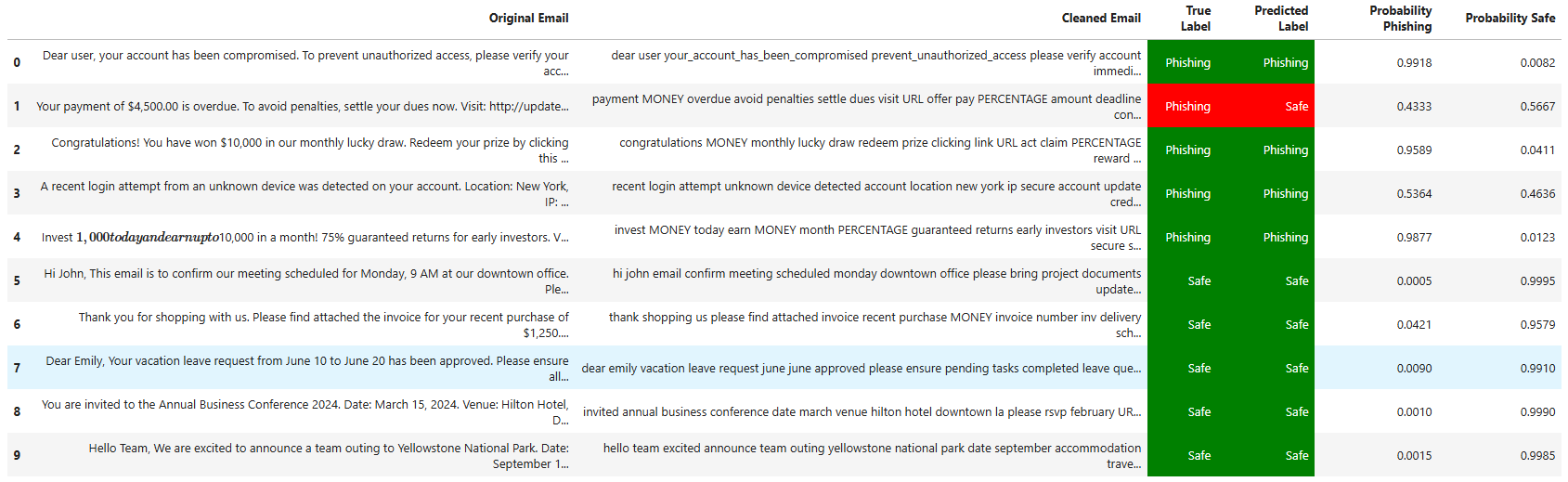


Table 10: LSTM Model Classification

**Analysis of Naive Bayes, GRU, and LSTM**

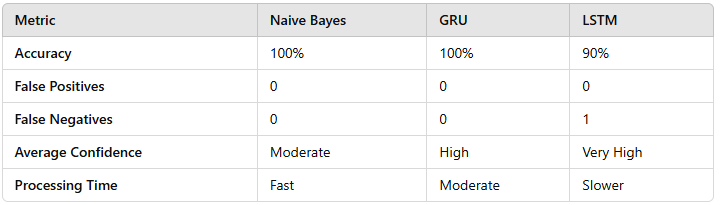


Table 11: Comparative Table of Naive Bayes, GRU, LSTM

**Observations**

Naive Bayes:

* Achieved 100% accuracy, demonstrating its strength in this dataset. However, it relies heavily on keyword frequency, which may not generalize well to nuanced real-world cases.
* Offers excellent speed and simplicity, making it a reliable choice for resource-constrained environments.

GRU:

* Matched Naive Bayes in accuracy but excelled in confidence scores, particularly for more complex phishing patterns.
* Its ability to understand context makes it more adaptable to varied datasets, though at a higher computational cost.

LSTM:

* While LSTM demonstrated strong performance, the single misclassification highlights a potential sensitivity to emails lacking clear phishing patterns.
* Computationally more expensive but highly suitable for datasets with complex and obfuscated phishing emails.

**Conclusion**

* **Naive Bayes**: A simple, efficient baseline model that performs well for explicit phishing patterns. Best suited for environments where speed and resource efficiency are critical.
* **GRU**: Balances high accuracy and strong confidence with moderate resource requirements, making it ideal for real-time applications with complex data.
* **LSTM**: Excels in handling intricate and obfuscated email structures but requires careful tuning and greater computational resources.

Both Naive Bayes and GRU achieved perfect accuracy in this dataset, with GRU offering better adaptability for real-world scenarios. Future work could explore hybrid models or ensemble techniques to leverage the strengths of all three approaches.

### Front End and Client Integration Result

The successful integration of the front end with the phishing detection system and client email platform demonstrates the seamless functionality of the solution. This section highlights how real emails fetched from the Microsoft 365 mailbox were processed, classified, and displayed on the system’s **Detail Page**, **Dashboard**, and the user's email inbox. The results validate the system's ability to provide real-time insights and classification updates.

**Display on the Detail Page**

* Grid View:
  + The Detail Page successfully displayed the classified emails in a tabular format, including:
  + Original Email: Shows the raw content of the email fetched from the user's mailbox.
  + Cleaned Email: Displays the preprocessed email content used for classification.
  + True Label and Predicted Label: Indicates the actual and predicted classifications of each email.
  + Classification Probabilities: Provides the confidence scores for "Phishing" and "Safe" classifications.
* Dynamic Updates: The system dynamically refreshed the grid when new emails were fetched and processed

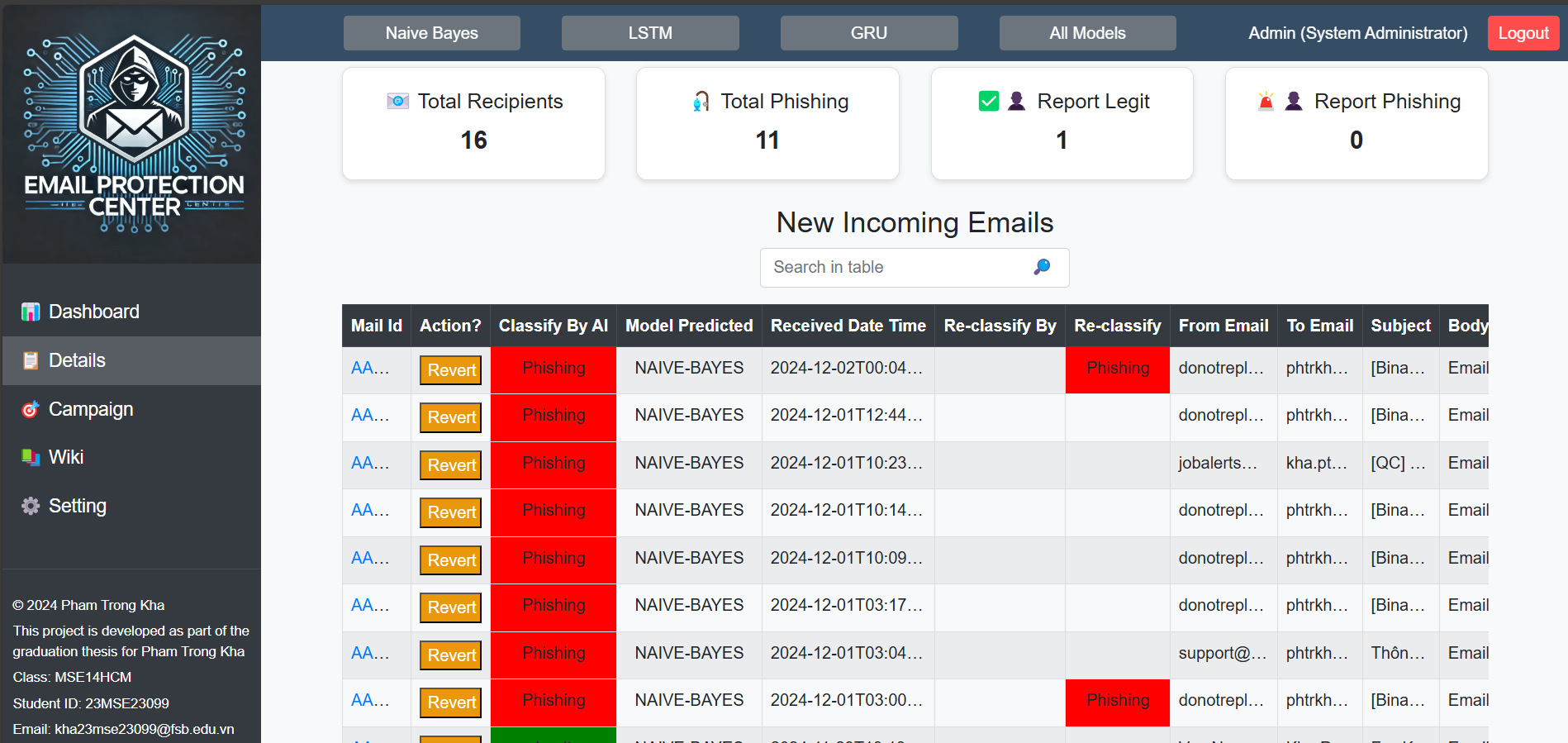


Figure 21: Email Protection Center detail page

**Analysis of Visualization on the Dashboard**

The **Dashboard** provides a centralized view of the phishing detection system's performance through various summary metrics, visualizations, and insights. The data visualized highlights key trends and statistics related to email classifications and user interactions, offering an intuitive way for administrators to monitor the system's effectiveness.

**Model Selection Feature**

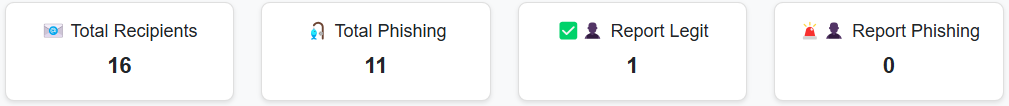


The Model Selection Panel allows users to dynamically choose the machine learning model for email classification. Available models include:

* Naive Bayes: Lightweight and fast for basic classifications.
* GRU: Balances accuracy and efficiency, ideal for sequential data.
* LSTM: Handles complex patterns with high accuracy.
* All Models: Runs all models for comparison or ensemble predictions.

The selected model is highlighted (e.g., GRU in the image) and updated in real-time via the backend. This feature empowers users to adapt the system to specific requirements, enabling flexibility, experimentation, and enhanced performance.

**Summary Cards**



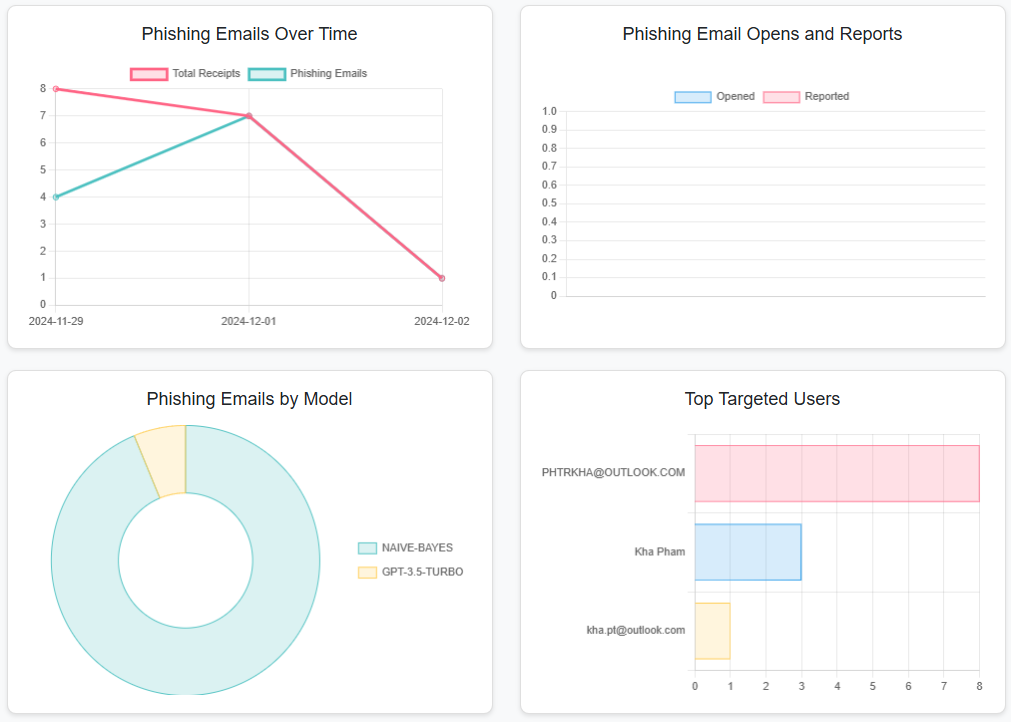
The summary cards at the top of the dashboard offer a quick overview of essential metrics:

* Total Recipients: Displays the total number of emails have been processed.
* Total Phishing: Indicates the number of emails classified as phishing.
* Report Legit: Emails that were incorrectly flagged and reported by users as legitimate.
* Report Phishing: Reflects phishing emails flagged manually by users.

Insights: These metrics provide a snapshot of system activity and user feedback, highlighting the system’s performance in classifying phishing emails and tracking user actions.

**Dashboard Chart Functionality Overview**

The charts on the Dashboard serve as critical tools for visualizing and understanding the system's performance and phishing activity patterns. Each chart focuses on a specific aspect of the system, aiding administrators in tracking, analyzing, and responding to phishing threats effectively:



Phishing Emails Over Time:

* + Tracks the daily trend of phishing emails compared to the total received emails.
  + Helps identify peaks in phishing activity, evaluate the system’s ongoing effectiveness, and uncover patterns in attack behavior.

Phishing Email Opens and Reports:

* + Monitors user interaction with phishing emails by tracking the number of opened and manually reported emails.
  + Provides insights into user awareness and trust in the system’s classifications.

Phishing Emails by Model:

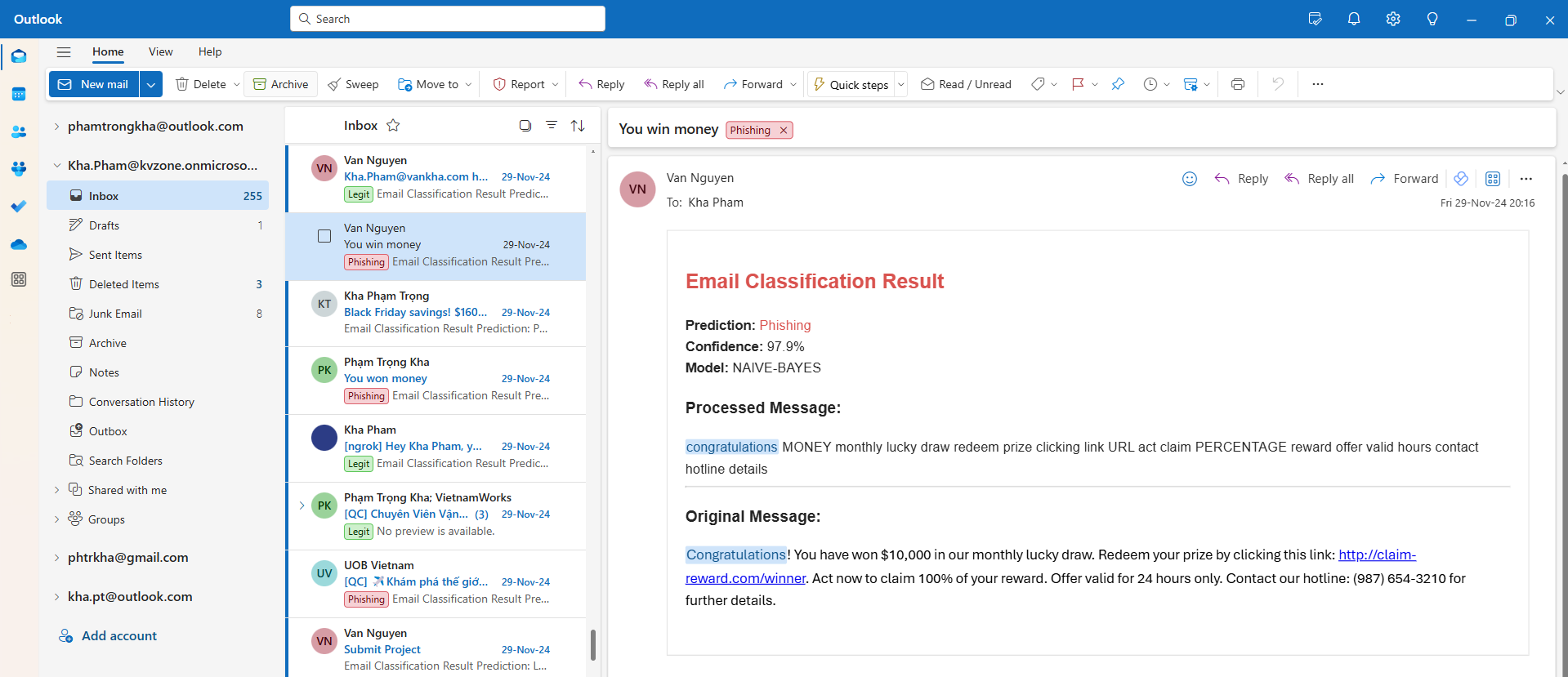
* + Shows the distribution of phishing emails detected by each machine learning model.
  + Enables evaluation of model contributions and effectiveness, guiding decisions on model usage and improvement.

Top Targeted Users:

* + Highlights users who received the most phishing emails.
  + Assists in identifying vulnerable accounts and implementing targeted protective measures.

These charts collectively provide a comprehensive view of system performance, user engagement, and phishing trends, ensuring administrators can make informed decisions to improve security.

**Display in the User's Email Inbox**



* Email Updates:
  + The system successfully updated emails in the user's Outlook inbox, tagging them with the appropriate classification:
    - Subject Tags: Added labels like "[Phishing]" or "[Safe]" to email subjects.
    - Category Assignment: Marked emails with color-coded categories for better visual differentiation in the inbox.

**Observations**

* Real-Time Functionality:
  + The system effectively handled real-time email fetching and classification updates, reflecting changes immediately on the front end and in the user’s email client.
* User-Friendly Design:
  + The intuitive grid on the Detail Page and visually rich Dashboard made it easy for users to track and understand phishing trends and email statuses.
* Reliability:
  + All classified emails were accurately displayed on the system and updated in the user's inbox without delays or errors.

## **Discussion**

The phishing detection system presented in this project demonstrates a robust and functional solution for identifying and managing phishing emails. The system integrates advanced machine learning models, Azure Graph API, and a user-friendly front end to provide real-time classification, visualization, and actionable insights. However, there are several areas where improvements and optimizations can be made:

**Local Deployment**:

Currently, the system is deployed and tested locally, which limits its scalability and accessibility. While this setup was sufficient for the scope of this project, it highlights the need for a cloud-based deployment for broader usability and better performance.

**Model Performance**:

The models (Naive Bayes, GRU, and LSTM) have shown excellent performance in identifying phishing emails. However, their performance could be further enhanced by integrating more advanced architectures like BERT combined with LSTM or transformer-based models to better handle complex and obfuscated phishing strategies.

**Language Limitation**:

The current system supports only English emails. This limitation restricts its applicability in multilingual environments. Expanding support for other languages is essential to make the system more inclusive and globally relevant.

**Real-Time Processing**:

While the system processes emails in real-time using webhook notifications, handling a higher volume of simultaneous emails could strain the current architecture. Optimizing the backend to support large-scale concurrent email processing is critical for enterprise use cases.

**Advanced Email Content Analysis**:

The system primarily focuses on textual data. Incorporating advanced techniques like URL path analysis, malware detection, and attachment inspection can significantly improve its phishing detection capabilities.

# Chapter 5: Conclusion and Future Work



## **Conclusion**

This project presents a comprehensive phishing detection system that leverages advanced machine learning models, real-time email processing, and an intuitive user interface to tackle the growing challenge of email-based cyber threats. The integration of the Naive Bayes, GRU, and LSTM models with Azure Graph API and a responsive front end ensures a robust and functional solution for identifying and managing phishing emails.

The system successfully demonstrates the ability to classify emails accurately, provide real-time updates, and present actionable insights through detailed visualizations and user-friendly tools. While the current implementation operates locally, it serves as a solid foundation for further enhancements, including cloud deployment, advanced model integration, multilingual support, and improved scalability to handle larger datasets and concurrent email traffic.

This work highlights the potential of combining machine learning, modern APIs, and thoughtful system design to address real-world cybersecurity challenges.

## **Future Work**

To address the current limitations and expand the system's functionality, the following future enhancements are proposed:

**Cloud Deployment**:

* Transition the system to a cloud-based infrastructure to improve scalability, reliability, and accessibility.
* Leverage cloud services like AWS, Azure, or Google Cloud to handle larger email volumes and provide seamless integration for multiple users and organizations.

**Integration of Advanced Models**:

* Incorporate advanced models such as BERT combined with LSTM to enhance the detection of complex phishing patterns.
* Explore ensemble learning techniques to combine the strengths of different models for better accuracy and robustness.

**Multilingual Support**:

* Extend the system to support multiple languages beyond English, using techniques like multilingual embeddings or pretrained models like mBERT.
* This would make the system applicable in diverse environments and industries.

**Advanced Analysis Techniques**:

* Integrate advanced tools to analyze URLs, detect malware in email attachments, and assess the overall risk of emails.
* Apply heuristic and sandboxing techniques to evaluate suspicious files and links dynamically.

**Scalable Architecture**:

* Redesign the system architecture to efficiently handle high volumes of emails and concurrent requests.
* Incorporate message queues (e.g., RabbitMQ or Kafka) and serverless functions to ensure the system remains responsive under heavy workloads.

**User Behavior Integration**:

* Develop mechanisms to incorporate user feedback into the system to improve model accuracy dynamically.
* Implement active learning techniques where mislabeled emails are used to retrain models periodically.

**Improved Security Measures**:

* Strengthen the system's security by incorporating email spoofing detection, anti-phishing frameworks, and encrypted data handling.
* Ensure compliance with industry standards like GDPR for email handling.

**Dashboard Enhancements**:

* Expand the dashboard to include predictive analytics and trend forecasting for phishing activity.
* Provide detailed user-specific insights to help administrators take proactive measures.

References

[1] Verizon, "2024 Data Breach Investigations Report," 2024. [Online]. Available: [https://www.verizon.com/business/resources/reports/dbir/.](https://www.verizon.com/business/resources/reports/dbir/)

[2] M. Tracy, W. Jansen, and K. Scarfone, "Guidelines on Electronic Mail Security," *NIST Special Publication 800-45 Version 2*, National Institute of Standards and Technology, 2007.

[3] Atawneh, S.; Aljehani, H. "Phishing Email Detection Model Using Deep Learning." *Electronics 2023, 12*, 4261. <https://doi.org/10.3390/electronics12204261>.

[4] Deloitte, " Cyber considerations amid a global pandemic," 2022. [Online]. Available: <https://www.deloitte.com>.

[5] N. Abdullah, "Phishing Email Dataset," Kaggle, 2023. [Online]. Available: <https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset>.

[6] "Incremental learning," *Wikipedia*, Dec. 30, 2024. [Online]. Available: <https://en.wikipedia.org/wiki/Incremental_learning>.

[7] "Create subscription," *Microsoft Learn*, Oct. 28, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/graph/api/subscription-post-subscriptions?view=graph-rest-1.0&tabs=http>.

[8] " Microsoft Graph REST API v1.0 endpoint reference," *Microsoft Learn*, July. 16, 2024. [Online]. Available: <https://learn.microsoft.com/en-us/graph/api/overview?view=graph-rest-1.0&preserve-view=true>.

[9] J. Patel, "URL-based Heuristic Detection of Phishing Emails," *University of Victoria*, 2018. [Online]. Available: <https://dspace.library.uvic.ca>.

[10] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Montreal, QC, Canada, 2006, pp. 581-590. [Online]. Available: <https://people.eecs.berkeley.edu/~tygar/papers/Phishing/why_phishing_works.pdf>

[11] G. Xiang and J. Hong, "A hybrid phishing detection approach by identity discovery and keywords retrieval," in *Proc. 18th Int. Conf. World Wide Web (WWW)*, Madrid, Spain, 2009, pp. 571-580. [Online]. Available: <https://dl.acm.org/doi/10.1145/1526709.1526787>

[12] A. Mandadi, S. Boppana, V. Ravella, R. Kavitha, " Phishing Website Detection Using Machine Learning," *IEEE Access*, vol. 8, pp. 204603-204616, Nov. 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9824801>

[13] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, "Client-side defense against web-based identity theft," in *Proc. 11th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, Feb. 2004, pp. 1-16. [Online]. Available: <https://crypto.stanford.edu/SpoofGuard/webspoof.pdf>

[14] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091-2121, Fourth Quarter 2013. [Online]. Available: <https://ieeexplore.ieee.org/document/6497928>

[15] Microsoft, "SmartScreen Filter Overview," 2021. [Online]. Available: <https://learn.microsoft.com/en-us/defender-office-365/anti-spam-protection-about>.

[16] Wikipedia contributors, "Naive Bayes spam filtering," Wikipedia, The Free Encyclopedia, [Online]. Available: <https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering>.

[17] H. A. Mohammed, T. A. Sulaiman, and A. S. Mahmood, "Detection of Phishing Websites Using Naive Bayes Algorithm," *IEEE Int. Conf. Computer Applications & Information Security (ICCAIS)*, Riyadh, Saudi Arabia, 2022. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/10561829>.

[18] M. Al-Kasassbeh, I. M. Qattash, and A. Al-Madi, "Detection of Phishing Websites Using Naïve Bayes Algorithm," *International Journal of Recent Research and Review (IJRRR)*, vol. 11, no. 4, pp. 45-55, Dec. 2022. [Online]. Available: <https://www.ijrrr.com/papers11-4/paper6-Detection%20of%20Phishing%20Websites%20Using%20Na%C3%AFve%20Bayes%20Algorithms.pdf>.

[19] X. Zhang, Y. Fang, and W. Zhao, "Machine Learning Algorithms for Phishing Detection: A Comparative Study," *IEEE Trans. Computational Social Systems*, vol. 8, no. 5, pp. 993-1004, Oct. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9930531>.

[20] K. Cho et al., "Gated recurrent unit," *Wikipedia*, Dec. 2024. [Online]. Available: <https://en.wikipedia.org/wiki/Gated_recurrent_unit>.

[21] S. A. Abbas et al., "Evaluating machine learning and deep learning approaches for phishing URL detection: A systematic review and future directions," *Academia.edu*. [Online]. Available: <https://www.academia.edu/125145389/Evaluating_Machine_Learning_and_Deep_Learning_Approaches_for_Phishing_URL_Detection_A_Systematic_Review_and_Future_Directions>.

[22] M. Al-Khateeb and H. A. Alshehri, "Hybrid detection system using CNN and GRU for phishing website detection," *SN Computer Science*, vol. 5, no. 3, 2024. [Online]. Available: <https://link.springer.com/article/10.1007/s42979-024-03282-6>.

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://www.bioinf.jku.at/publications/older/2604.pdf>.

[24] X. Zhang et al., "Phishing Detection Using LSTM Networks for Large Email Datasets," *Proc. IEEE Int. Conf. Computational Intelligence*, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9034081>.

[25] M. Al-Khateeb and H. A. Alshehri, "Hybrid detection system using CNN and LSTM for phishing website detection," *SN Computer Science*, vol. 5, no. 3, pp. 1-12, 2024. [Online]. Available: <https://link.springer.com/article/10.1007/s42979-024-03282-6>.

[26] S. A. Abbas et al., "Evaluating machine learning and deep learning approaches for phishing URL detection: A systematic review and future directions," *Academia.edu*, 2023. [Online]. Available: <https://aircconline.com/ijcnc/V15N1/15123cnc02.pdf>.

[27] F. Arden, "Phishing Email Detection Thesis," *Kaggle*, 2023. [Online]. Available: <https://www.kaggle.com/code/farelarden/phishing-email-detection-thesis>.

[28] CyberSecTony, "Phishing Email Detection v2.0," *Hugging Face Datasets*, 2023. [Online]. Available: <https://huggingface.co/datasets/cybersectony/PhishingEmailDetectionv2.0>.

[29] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Pearson, 2024.