International Journal of Electrical and Computer Engineering (IJECE)
Vol. 16, No. 3, June 2026, pp. 1608~1625
ISSN: 2088-8708, DOI: 10.11591/ijece.v16i3.pp1608-1625

A multi-modal framework for improving the accuracy of phishing email detection

Lamees Mohamed Faraj[1], Sayed Abdel-Gaber[2], Hanan Fahmy[2]
[1]School of Computer Science, Canadian International Collage, Cairo, Egypt
[2]Department of Information Systems, Faculty of Computers and Artificial Intelligence, Capital University, Helwan, Egypt

Article Info

Article history:

Received Dec 1, 2025
Revised Feb 19, 2026
Accepted Apr 26, 2026

Keywords:

Attachment-based analysis
Email security
Machine learning
Multi-model detection
Neural network
Phishing email detection

ABSTRACT

Phishing emails continue to pose a significant cybersecurity threat, particularly through the increasing use of malicious attachments to evade traditional text-based detection systems. Most existing approaches focus primarily on email content, creating a blind spot in attachment-aware phishing detection. This paper proposes a multi-modal phishing email classification model that integrates email header features, body text analysis, and attachment inspection within an ensemble learning framework. Independent machine learning classifiers are employed for each email component, and a majority voting mechanism is used to determine the final classification decision. The proposed model is evaluated using publicly available email and attachment datasets that are combined to simulate attachment-bearing phishing emails. Experimental results demonstrate strong detection performance across multiple evaluation metrics. Nevertheless, the study acknowledges the limitation of using synthetically paired email bodies and attachments, which may not fully capture real-world semantic relationships. The findings highlight the importance of incorporating attachment-aware analysis into phishing detection systems and provide a foundation for future research on semantic consistency modeling and transformer-based architectures.

Corresponding Author:

Lamees Mohamed
School of Computer Science, Canadian International Collage
Cairo, Egypt
Email: Lamees_m_frj@cic-cairo.com

1. INTRODUCTION

Phishing emails continue to pose a major cybersecurity threat, with attacks increasingly leveraging malicious attachments to bypass traditional text-based detection systems. Recent reports indicate that over 90% of cyberattacks in 2023 involved some form of social engineering, including phishing campaigns targeting both individuals and organizations (Microsoft Security Intelligence Report, 2023). Despite numerous advances in email security, current detection systems primarily focus on textual content, creating a blind spot in attachment-aware phishing detection. This study addresses this gap by proposing a multi-modal classification model that integrates header features, body text analysis, and attachment inspection [1]. Phishing is a cyberattack that combines social engineering techniques with technical manipulation to obtain sensitive user information. In this type of attack, the adversary sends deceptive emails designed to mislead recipients into revealing confidential data. This information may be exposed through different methods, such as clicking on harmful links that lead to fraudulent websites where login details are entered, directly sharing the data, or downloading attachments embedded with malicious software [2]. Phishing often exploits weaknesses in system processes, and from the end user's perspective, defending against it largely depends on

their knowledge and awareness, which can vary widely among individuals [1]. Even with a well-secured system, an unaware user may be tricked into revealing personal or financial details, which can further jeopardize the organization's security by granting unauthorized access to its sensitive data. McAfee estimates that cybercrime-related losses cost the global economy approximately one trillion US dollars annually [1], [3]. Overall, phishing continues to be a significant threat affecting both individuals and businesses [4].

Recently, phishing attackers have increasingly relied on malicious email attachments, such as PDF documents, images, and compressed files, to bypass traditional text-based detection systems, revealing a critical blind spot in many existing phishing detection approaches [4]. A study conducted by Birthriya and Jain on phishing email detection indicates that phishing attacks most frequently target specific sectors, with the SaaS industry accounting for 34.7%, followed by financial institutions at 18%, payment services at 11.8%, social media platforms at 10.8%, and e-commerce businesses at 7.5% [5]. According to recent cybersecurity reports, phishing remains one of the most prevalent attack vectors worldwide, with a continuous increase in email-based attacks leveraging social engineering and malicious attachments. Reports published in 2023 and 2024 indicate that phishing emails continue to account for a significant portion of successful cyberattacks across both personal and organizational environments [4]. Given the vast volume of potentially malicious emails, along with their increasing diversity, there is a pressing need for solutions that enhance the tools and techniques email service providers use to detect harmful emails [4]. Email messages typically consist of three main components:

a. The message envelope, which represents the email's electronic format.
b. The message header, which contains details such as the email subject line and sender/recipient information.
c. The message body, which includes text, images, and file attachments.

Figure 1 illustrates an example of the standard structure of a malicious spam email. Such emails often employ social engineering techniques in their subject line or body text to deceive recipients into interacting with the message. For example, the recipient may be instructed to download a compressed file, pretending to retrieve a fake undelivered item. These files are intentionally misleading and often use multiple file extensions to obscure their malicious nature [6].

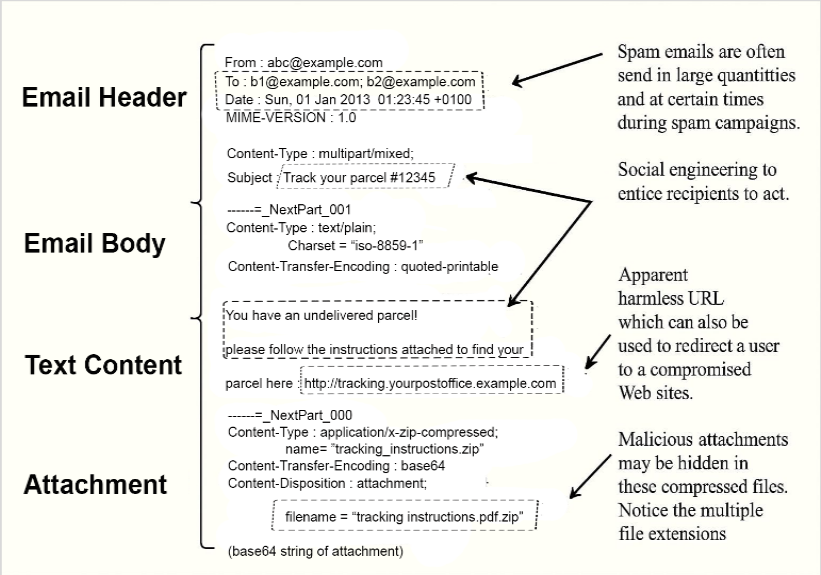


Figure 1. An example of (fake) spam email with a potential malicious attachment and URL [7]

The goal of this compressed file in Figure 1 is to hide malware executable from virus scanners run by or applied by mail servers. In this example, the URL acts as a secondary method to spread harmful material. Similar to malicious files, malicious URLs can hide a harmful website (such as *example.com*) by adding subdomains that resemble a trusted and secure website (*e.g.*, *tracking.yourpostoffice*). This example also illustrates a possible spam template where URLs or attachments may have different names but still serve the same malicious purpose [8].

Despite the effectiveness of existing machine learning-based phishing detection techniques, most proposed solutions primarily focus on analyzing email headers or body text, while the analysis of malicious

attachments is often treated as a secondary or independent task [8], [9]. Various techniques are employed for filtering unsolicited emails, including knowledge-based methods, learning-based approaches, clustering techniques, heuristic procedures, and others. However, these techniques often struggle to prevent bypass attacks. To address this limitation, this paper introduces the proposed phishing email classifier model (PECM) designed to classify email segments as either spam or legitimate (ham). The model identifies gaps in existing email filtering systems and analyzes phishing email links and malicious attachments.

Spam emails containing malicious attachments or URLs aim to deceive recipients into opening attachment or clicking on links. The subject lines and body text of these emails are crafted to tempt or alarm the recipient into responding to the disguised malicious content. Scanning URLs and attachments against virus scanners or blacklists, often revealing the nature and scope of the malicious content in these harmful emails. However, these scanning procedures usually require external resources, which can be costly to compute and hard to maintain. While many spam detection and filtering systems ty to adapt to changing spamming strategies, they are often too inflexible to effectively handle the evolving nature of spam emails [8]. There has been limited research on detecting malicious content—specifically attachments or URLs—in spam emails. This work addresses that gap by focusing on classifying and phishing emails based on their URLs and attachments. Unlike similar approaches, the proposed PECM is self-contained and does not rely on external resources such as blacklists. Given the rapid adoption of new tactics by attackers, it is crucial to improve the accuracy of phishing email detection systems and to mitigate bypass attempts, as the content of malicious attachments and phishing links frequently changes. Therefore, this paper aims to improve phishing email detection accuracy by proposing a classification model that jointly analyzes email headers, body content, and attachments using machine learning and ensemble-based decision mechanisms.

The main contributions of this paper can be summarized as follows:

a. We propose a phishing detection model that analyses email attachments in addition to headers and body content.
b. We address a critical blind spot in existing systems by incorporating malicious attachment features.
c. We evaluate the model on a benchmark dataset, demonstrating improved detection accuracy compared to baseline methods.
d. We present a modular framework that can be integrated into existing email security systems.

The rest of this paper is organized as follows. The section titled “related work” introduces the key definitions and reviews prior studies in this area. In “the proposed phishing email classifier model (PECM),” the adopted methodology and the proposed model are described. The “evaluation and results” section provides an analysis of the outcomes to measure the performance of the proposed approach. This is followed by a discussion of the study’s limitations. Finally, the “conclusion” summarizes the main findings and highlights potential directions for future research.

2. RELATED WORK

This section summarizes related studies that highlight methodologies based on machine learning for spam detection. The following related studies are divided into three subsections. First, the studies that focus on email spam detection. Second the studies that focus on malicious attachment detection. Finally, the studies that focus on malicious URL detection [10].

2.1. Email spam filtering and detection

This subsection reviews methods for detecting malicious emails and introduces a taxonomy of these techniques, as illustrated in Figure 2. The taxonomy is divided into two levels: Email components needed for feature extraction, such as the header, body, or attachment, are identified by Layer 1. The accuracy of classification is influenced differently by each component. Layer 2: differentiates between common machine learning (ML) and deep learning (DL) techniques for identifying fraudulent and benign emails [11].

Beaman and Isah [12] focused on anomaly detection in email headers, by using features such as missing fields, blacklisted SMTP servers, and time zones. The one-class support vector machine (SVM) achieved 87% accuracy in detecting phishing emails, while a stacked ensemble (random forest (RF), support vector machine (SVM), multilayer perceptron (MLP) dan K-nearest neighbors (KNN)) achieved 97% accuracy in phishing detection.

AbdulNabi and Yaseen [13] examined the effectiveness of word embedding techniques in detecting spam emails. They enhanced the performance of bidirectional encoder representations from transformers (BERT), a pre-trained transformer-based model, to classify emails into spam and legitimate (ham) categories. Owing to its attention mechanism, BERT is capable of capturing contextual relationships within text, which contributes to improved classification performance. For comparison purposes, a baseline deep neural network (DNN) architecture was implemented, consisting of two fully connected layers along with a bidirectional

long short-term memory (BiLSTM) layer. Additionally, the study evaluated traditional machine learning approaches such as naive Bayes (NB) and KNN. The proposed approach achieved a peak accuracy of 98.67% and an F1-score of 98.66%. These results were obtained using two publicly available datasets—one designated for training and the other for testing—to assess the model's generalization capability and robustness when handling unseen data.

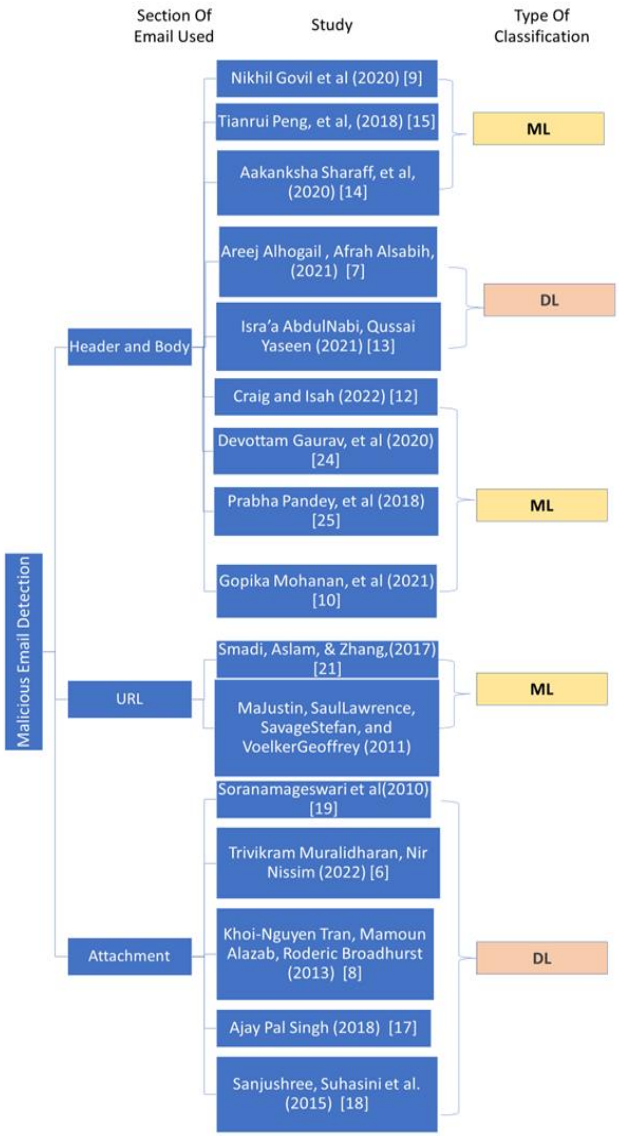


Figure 2. Taxonomy of malicious email detection studies

Sharaff and Srinivasarao [14] analyzed the correlation between the words used in email subject lines and their corresponding message content to determine the nature of the emails. They applied a feature selection process to identify the most significant terms from the entire vocabulary, which were then utilized to construct N-gram features. The effectiveness of these N-gram representations was evaluated using four machine learning classifiers: decision tree, multinomial NB, RF, and linear SVM. The experimental evaluation was conducted using the Ling-Spam dataset.

Peng *et al.* [15] concentrated on using semantic analysis to identify malicious intent in email assaults by examining the natural language text. A sizable benchmark set of phishing emails was used to show the efficacy of this strategy.

Alhogail and Alsabih [7] proposed a phishing email classification model based on deep learning techniques. Their approach enhances detection accuracy by applying natural language processing (NLP) to analyze email content and incorporating a graph convolutional network (GCN).

2.2. Classification of malicious attachments

Muralidharan and Nissim [6]. proposed an automated malicious email detection framework using deep ensemble learning, that analyzes all segments of an email. Their approach demonstrated that an ensemble framework of deep learning classifiers, each trained on different portions of an email, can offer better generalization than traditional methods that only analyze specific sections of an email.

Listík *et al.* [16] developed a model architecture that is based on convolutional neural networks (CNN), which have demonstrated impressive picture categorization performance. They evaluated this architecture on two publicly accessible datasets and were able to classify email images as spam with state-of-the-art performance. On both the Princeton and Dredze datasets, the suggested CNN architecture achieved an F1-score of 96% and 99%, respectively. Additionally, using the Email.cz image spam v1 dataset, it obtained an F1-score of up to 96%.

Tran *et al.* [8] proposed a number of novel features for identifying dangerous URLs and attachments in spam emails. According to their hypothesis, the best basis for detecting spam emails with harmful attachments or URLs is the text content in the email subject and body. Singh *et al.* [17] aimed to enhance the classification results by applying neural networks and DNN to the spam image classification problem.

Sanjushree and Meena [18] used particle swarm optimization (PSO) and SVM on three textual features and ten metadata features to categorize image spam. On the Dredze dataset, which contained 300 training images and 380 test images, this combination of methods produced an accuracy of 90%. With client and server models, the authors additionally employ cluster-based filtering, claiming a 99% accuracy rate. Soranamageswari *et al.* [19] achieved 92.82% accuracy using a color feature-based backpropagation neural network. They also tried dividing the image into pieces and used those blocks as features; however, this approach only produced an accuracy of 89.32%.

2.3. Classification of malicious URLs

Research on detecting malicious URLs has expanded beyond spam email filtering, driven by the extensive use of URLs across web content and digital communications. Although blacklisting remains a commonly used approach to restrict access to harmful links, its effectiveness depends on previously identifying and cataloging such URLs [20].

Smadi *et al.* [21] proposed a neural network–based approach that utilizes four key components of an email: the message body (nine features), HTML structure (10 features), embedded URLs (12 features), and header information (19 features). They also introduced feature evaluation and reduction (FEaR), a tailored feature selection method designed to identify the most suitable subset of features based on the dataset's size and characteristics.

Ameeno *et al.* [22] were among the early researchers to address phishing detection through the analysis of malicious links. Their work incorporated various features, including IP address characteristics, WHOIS records, domain-related attributes, blacklist inclusion, geographic location, and connection speed. The study further revealed that online learning techniques outperformed batch learning approaches, with the confidence-weighted online method achieving better results than several SVM variants.

More recently, phishing email detection has benefited from the adoption of transformer-based architectures such as BERT [23]. These models excel at capturing contextual and semantic information in text through deep bidirectional representations, often surpassing traditional machine learning methods in performance. Nevertheless, their high computational demands and extended training times can pose challenges for deployment in real-time systems or environments with limited resources [24].

3. METHOD

This section outlines the proposed PECM, and the steps involved in conducting the model, the proposed PECM employs machine learning algorithms to analyze all email components—body, header, and attachments. Enhancing phishing detection accuracy to prevent unwanted and potentially harmful emails from reaching the user's inbox. The process of building the proposed PECM involves three main steps. First step, data from both phishing and legitimate emails are collected and prepared for pre-processing stage. Next, the proposed PECM employs bag-of-words (BoW) and TF-IDF methods to extract meaningful features. Then, the proposed PECM is developed using machine learning algorithms, with the training data being fed into the classifier to construct the model and prepare it for evaluation. The final phase involves testing the proposed PECM with the testing data to validate the model's performance.

In this work, the proposed phishing detection model is primarily designed to operate as a self-contained system based on machine learning techniques. However, VirusTotal is utilized only during the dataset labeling and validation phase, not during the real-time detection process. Therefore, the model does not rely on any external third-party services during deployment, ensuring independence, efficiency, and faster

response time in real-world applications. While the proposed PECM analyzes each email component separately, the final classification decision is made using a majority voting ensemble mechanism. Specifically, separate classifiers are trained for the header, body, and attachment features, and each classifier produces a binary output (phishing or legitimate). The final decision is determined by the class that receives the majority of votes.

$$\begin{Bmatrix} if\ a\ majority\ exists & ,Majority\ Y \\ if\ tie\ occurs, priority: Body > Attachments > Header & ,Priority\ \ Y \end{Bmatrix} = Final\ Y$$

This ensemble approach was chosen due to its ease of implementation, clear interpretability, and effectiveness in integrating diverse feature sets. The proposed PECM is shown in Figure 3, and a detailed description of each step is provided in the following subsection.

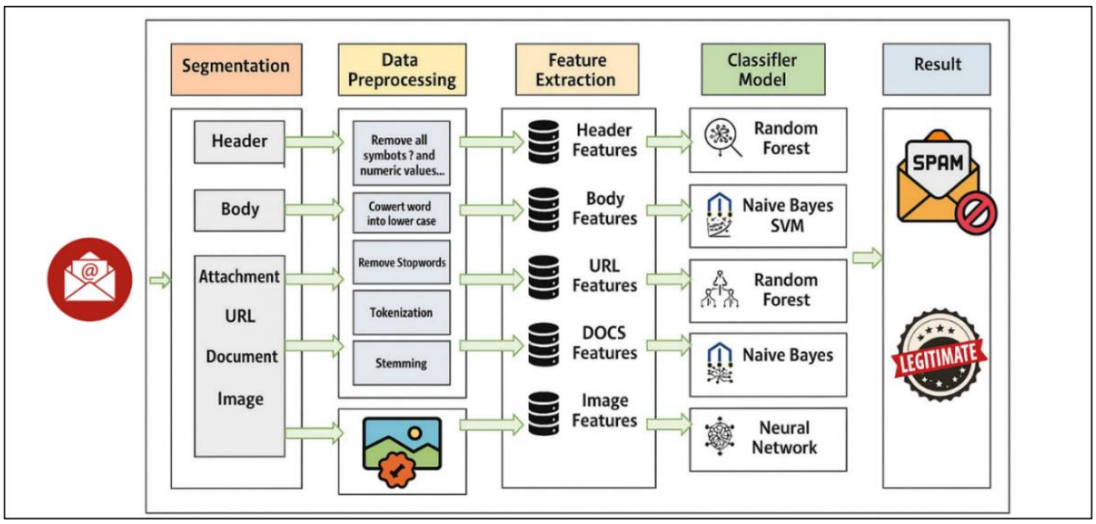


Figure 3. Phishing email classifier model (PECM)

3.1. Data preprocessing

In order to ensure that the data is clean and appropriate for analysis, data preparation is an essential step in developing the suggested PECM. Unnecessary words and characters are removed during this process. Once cleaned, formal feature extraction techniques can be applied. This study utilized several preprocessing techniques, including tokenization, stop-word removal, and email cleaning. Each method is described below:

3.1.1. Removal of whitespace

Clean text is essential for machine learning models and typically consists of tokens or a list of words. This process involves converting raw text into a structured list of words. A simple approach to achieve this is to split the text based on whitespace, including spaces, newlines, and tabs. In Python, this can be done using the $split()$ function on the loaded string.

3.1.2. Removal of punctuation

To remove punctuation, a predefined string of punctuation characters is utilized. A for loop iterates through the provided text, and each character is checked for membership in the punctuation string. If the character is not punctuation, it is concatenated to an empty string. This results in a cleaned text string with punctuation removed.

3.1.3. Tokenization

Tokenization is the process of dividing each email into separate units, known as tokens, typically by splitting the text based on whitespace. In this work, this step is implemented using the $split()$ function.

3.1.4. Removal of stop words

Stop words are words that hold little significance in search queries or text analysis. Most search engines are programmed to disregard stop words. These words frequently occur in text and mainly include

conjunctions, articles, prepositions, and other functional words that support the structure of a sentence but lack standalone meaning. Table 1 illustrates sample of stop words.

Table 1. Sample of stop words

Words			
myself	theirs	let's	with
an	being	at	into
could	a	that's	between
what's	the	who's	against
about	but	at	before
through	for	by	during

3.1.5. Stemming

Stemming aims to reduce variations in inflected forms of a word by transforming them into their root form. There are two primary methods for completing this process: the Porter stemming algorithm and the dictionary-based method.

In the dataset used for this study, numerous images were present in various formats, with approximately 60-70% in GIF format. To enhance and preprocess the dataset, the GIF images were handled by extracting their initial frame and converting it into PNG format. The procedure followed to accomplish this is outlined below:

a. Conversion to PNG format

All *.GIF* images were transformed into .PNG format to ensure consistent processing. From each *.GIF* file, only the first frame was extracted and retained as the spam image, while the remaining frames that did not contain spam-related information were discarded.

b. Removal of corrupted files

Any corrupted image files were identified and removed from the dataset. To implement these steps, proof-of-concept scripts were executed. Custom Bash scripts were developed to execute these tasks and record the output at each stage, ensuring the dataset of spam images was properly cleaned and enhanced. Consequently, the target variable for labeling was encoded into a binary format, where spam was assigned a value of 1 and HAM was assigned a value of 0.

For this initial investigation, this study focuses on emails containing attachments or URLs to identify or anticipate harmful content. Additionally, significant terms that frequently appear in the spam email dataset are examined. Figure 4 illustrates the most common spam-related words from the combined datasets, which will help in identifying spam emails.

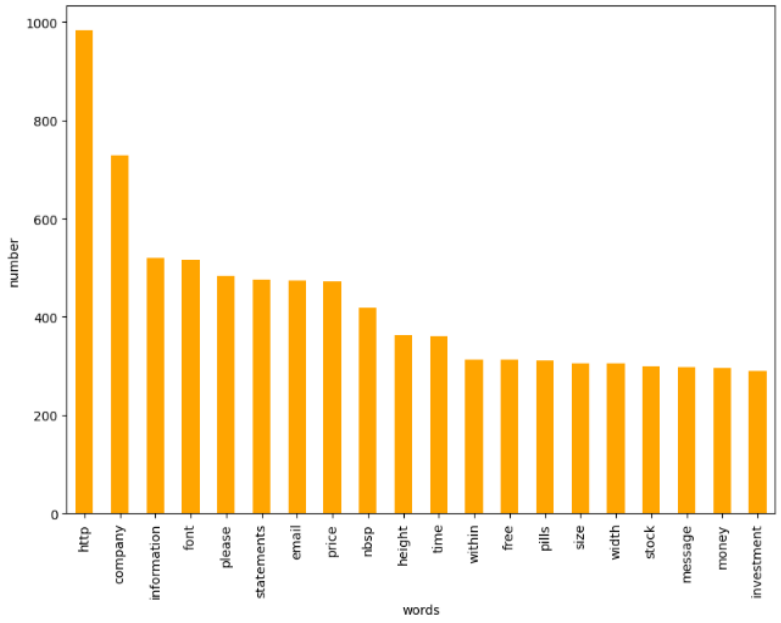


Figure 4. The most frequent spamming words

3.2. Feature extraction

After preprocessing, the TF-IDF technique is applied for feature extraction. Term frequency (TF) represents how frequently a word occurs in a document, whereas inverse document frequency (IDF) reflects how uncommon that word is across the entire corpus by relating the total number of documents to those that include the term. In this study, IDF is calculated using the "TfidfTransformer" module, which transforms text into a numerical format appropriate for machine learning algorithms. Typically, frequently occurring words are assigned values between 0 and 1, where lower values indicate that the term is common and carries less discriminative importance. This makes it possible for the algorithms to read the data. Equation (1), where (t, d) represents the term frequency (t) within a document (d), can be used to determine the TF-IDF:

$$tf - idf(t,d) = tf(t,d) \times idf(t) \quad (1)$$

where IDF is calculated by the (2), given *n* is the number of documents.

$$idf(t) = log(\frac{n}{idf(t)}) + 1 \quad (2)$$

The BoW approach is employed to extract features from textual data, after which different machine learning algorithms are applied to classify spam emails. The email content is converted into a vector representation using the *CountVectorizer* from the Scikit-learn library. This method records the frequency of each word appearing in the text, helping highlight important features for spam detection. Since machine learning algorithms cannot process raw text input directly, sentences and words in emails need to be converted into numerical representations or feature vectors. This process, known as feature extraction, involves transforming textual data into a form that machine learning algorithms can use. In this paper, the BoW model is employed as the feature extractor. It is one of the simplest and most widely used techniques for feature extraction in information retrieval. Rather than considering the order or position of words in sentences, the BoW model focuses solely on the frequency of word occurrences as the key feature for classification.

In our experiments, different feature extraction methods were applied depending on the nature of the email component. Specifically, the BoW method was employed for processing header fields and URL-related textual information, as these components benefit from simple token occurrence representation. In contrast, the term frequency–inverse document frequency (TF-IDF) method was applied to the body text and to textual content extracted from document attachments (*e.g.*, DOC, PDF) to better capture the importance of terms relative to the entire dataset. This separation of methods ensures that each component's unique characteristics are appropriately represented for the classification task.

To enhance semantic consistency between email components, an additional verification step was introduced to evaluate the contextual alignment between the email subject and the attachment filename. Keyword matching was performed after text normalization and tokenization, where common phishing-related terms (*e.g.*, invoice, payment, receipt, document) appearing in the subject line were compared against attachment filenames. This lightweight semantic consistency check contributes to detecting suspicious emails where misleading or mismatched attachments are employed.

3.2.1. Header feature class

The email header, which is typically not visible to end users, contains important metadata such as the transmission path, sending and receiving servers, timestamps, and message format. From this header, several informative features can be derived, including the existence of a message ID, identification of blacklisted terms in the subject line, whether the message is in plain-text format, consistency between the sender and return-path domains, discrepancies in IP addresses across successive Received fields, and verification using DomainKeys Identified Mail (DKIM). A subset of these attributes was extracted from the email header, and their detailed descriptions are provided in Table 2.

3.2.2. Body features class

Email content may contain URLs and be displayed in a variety of forms, including HTML and text. Consequently, the body class is divided into two subclasses: the URL subclass and the text subclass, which includes scripts and text content extracted from HTML.

a. Text feature class

Text analysis and an examination of the email's HTML content can be used to extract these features. Examples include the usage of difficult phrases in the text, the frequency or presence of tags like "href" and "onmouseover," and the occurrence of words that are on a blacklist or that elicit an action. The primary text that the reader may see is located in a separate area of the email body. This stage involves extracting the text

that the email recipient can see from the content. Some of these features were taken from the email body, and Table 3 lists their descriptions.

Table 2. Example of selected header features.

Feature	Feature Description
H01	This feature is used as a binary indicator to determine whether the domain names derived from the email sender match those from the message ID by comparing the sender's domain with the domain from the message ID.
H02	A binary feature that indicates if the content type of the email is set to HTML or Text.
H03	A binary feature that determines if the email's header specifies text or plain as the content type.
H04	Determines how many recipients are listed in the email header's “from”.
H05	Checks if the term “bank” appears in the email subject and returns “true” if it does or “false” otherwise.
H06	Checks if the term “debit” appears in the email subject and returns “true” if it does or “false” otherwise.
H07	Determines if the word “Fwd:” appears in the email subject and returns “true” if it does or “false” otherwise. .
H08	Checks to see if the word “Re:” appears in the email subject and returns “true” if it does or “false” otherwise.
H09	Identifies whether the word “verify” appears in the email subject and returns “true” if it does or “false” otherwise.
H10	Determines how many characters there are in the subject area and gives back the appropriate number.
H11	Determines the total word count in the subject field and provides the outcome.
H12	Checks to see if the email sender matches the “Reply To” field; if not, returns (false); otherwise, returns (true).
H13	Determines how many recipients are listed in the email header's “To:” field.
H14	Determines how many recipients are listed in the email header's “Cc:” field.
H15	Determines how many people are listed in the email header's “Bcc:” field.

Table 3. Example of selected body content features

Feature	Feature Description
B01	Checks to see whether there is an HTML form element in the email message.
B02	checks to see whether there is a JavaScript pop-up in the email.
B03	Determines how many pictures are used as hyperlinks.
B04	Determines whether the domain names have a corresponding reverse DNS record; if so, it returns (true); if not, it returns (false).
B05	A function that returns (true) if an email contains a onClick JavaScript event and (false) otherwise.
B06	Returns the email message's byte size.
B07	binary feature checks for the occurrence of the word “Dear” in the email body, assigning a value of true if it is present and false if it is not.
B08	Determines the email body's character count.
B09	Determines how many words are in the body of the email.
B10	determines how many of the specified keywords are in the body of the email. Account, click, identify, inconvenient, information, limited, log, minutes, password, recently, risk, social, security, service, and suspended are a few examples of these terms.
B11	Checks the email body for the word “suspension,” returning true if it is present and false if it is not.
B12	The phrase “verify your account” is checked for in the email text using a binary indicator, which returns true if it is present and false otherwise.

b. Email URL and DOC feature classes

Emails may contain a variable number of URLs, necessitating the aggregation of URL characteristics to prevent feature vectors with inconsistent lengths. Aggregated features collect specific information from every URL in an email and provide continuous or Boolean results. The number of URLs using a secure connection, the number of URLs having an IP address, and the number of URLs with domains other than the sender's address are a few examples. Tables 4 and Table 5 provide the complete list of features extracted from URLs and document attachments (DOCs), respectively. These features were used in the experimental setup of the proposed PECM for training and evaluation.

Table 4. Example of selected URL features

Feature	Feature Description
URL Length	Total number of characters in the URL
Number of Dots	Count of “.” characters in the URL
Presence of IP Address	Boolean flag indicating if the URL contains an IP address instead of a domain
Number of Subdomains	Count of subdomain segments in the URL.
HTTPS Usage	Boolean flag indicating if the URL starts with HTTPS
Special Characters Count	Number of special characters (@, #, ?, =, etc.)
Suspicious Keyword Presence	Boolean flag indicating if the URL contains suspicious keywords (*e.g.*, “login”, “verify”, “update”)
URL Entropy	Measure of randomness in the URL string
Redirection Count	Number of redirections the URL performs

Table 5. example of selected DOCS features

Feature	Feature Description
File Size (KB)	Size of the document in kilobytes
Number of Pages	Total page count in the document
Word Count	Number of words in the document
Embedded Links Count	Number of hyperlinks inside the document.
Macro Presence	Boolean flag indicating if macros are embedded
Suspicious Macro Keywords	Boolean flag for suspicious macro keywords (*e.g.*, "Shell", "CreateObject")
Suspicious Keyword Presence	Boolean flag indicating if the URL contains suspicious keywords (*e.g.*, "login", "verify", "update")
File Extension Type	File format extension (*e.g.*, .doc, .docx, .pdf)
Metadata Anomalies	Boolean flag if document metadata is suspicious
External Object Embedding	Boolean flag if the file contains embedded external objects

c. Email image feature class

Color, information, shape, texture, and noise characteristics are the five main categories into which image features are commonly divided. These categories provide a comprehensive representation of different visual properties that can be extracted from images for analysis. Table 6 presents a detailed overview of several representative feature groups along with their corresponding descriptions.

Table 6. Example of selected image features

Feature	Feature Description
Metadata Feature	This feature includes the image's height, width, bit depth, aspect ratio and compression ratio.
Color Feature	This feature encompasses the mean, skewness, variance, and entropy values of various image attributes, such as RGB color channels.
Texture Feature	This feature determines similarity and information of adjacent pixels.
Shape Feature	This Feature Measures the variations in intensity gradients within an image.
Noise Feature	This feature comprises the signal-to-noise ratio (SNR) and noise entropy.

3.3. Model training

In the model training and evaluation phases, we initially provided a general overview of the machine learning models used, along with a confusion matrix to illustrate classification performance. To enhance the reproducibility of our study and ensure the validity of our findings, we now present detailed configuration settings.

Python 3.10 and the Scikit-learn 1.3 library were used to implement each model. We employed an RBF kernel with "C=1.0" and "gamma=scale" for the SVM. The random forest classifier was set up with "criterion=gini," "max_depth=10," and "n_estimators=100." The Gaussian variant with default settings was used by the naïve Bayes model.

The dataset was split into 80% for training and 20% for testing. Additionally, a 5-fold cross-validation approach was used to ensure reliable evaluation. Each experiment was repeated ten times, and the results, including the confusion matrix, represent the average across these runs. This methodology was adopted to mitigate the impact of random variations and to provide stable performance metrics. In the proposed PECM, an email undergoing classification is divided into its three primary segments: the header, body and attachments. Each of these segments is trained using a distinct classifier.

3.3.1. Phishing email classification based on the header of an email

Email metadata, such as the sender's IP address, content type, recipient's email address, MIME version, and other attributes, is stored in the email header. The size of the header and the number of arguments it contains can vary significantly between emails.

Classifier: The random forest algorithm is used for header classification.

a. Random forest (RF)

RF algorithm is used to classify email headers. The random forest Classifier is an ensemble of decision trees, each varying in size and structure. During training, the data is sampled randomly, and at each tree node, subsets of input features are selected randomly. This randomization reduces correlations between individual trees, resulting in improved generalization and reduced ensemble error. By ensuring that the features used in tree splits differ across trees, the model becomes more robust and better suited for email header classification.

3.3.2. Phishing email classification based on the body of an email

Email bodies often contain plain text as well as additional elements such as URLs, HTML content and other links. The plain text typically includes the main message intended for the recipient, which may consist of simple words, images, JavaScript, and links to internal or external resources. HTML content is

primarily used for formatting and structuring the email's content, while URLs and links are commonly included to direct readers to other websites or resources. To detect potential risks, it's essential to focus on the context of the email's content. For instance, emails requesting information or prompting users to log into a portal—especially when accompanied by URLs—are often flagged as potentially harmful. Phishing attempts frequently mimic legitimate email content to bypass detection systems and appear trustworthy.

Classifier: For the email body classification task, the naïve Bayes (NB) and support vector machine (SVM) models were selected to provide a baseline for performance comparison against the proposed PECM. These models were trained and evaluated exclusively on the email body text features extracted using TF-IDF and BoW techniques. Their inclusion in the study was solely for comparative analysis, to benchmark the body-text classification capability of the proposed approach.

a. Naïve Bayes (NB)

The NB classifier is grounded in Bayes' Theorem and assumes that all features are independent of one another. In other words, the occurrence of a particular feature within a class is considered unrelated to the presence of any other feature, even if dependencies may exist in reality. Under this assumption, each feature contributes individually to the overall probability estimation. The probabilistic formulation of the model follows the independent feature assumption and is represented mathematically using the autonomous characteristic model, as illustrated in (3):

$$Bayes'\ Theorem:\ P(B/A) = \frac{Prob\ (A\ and\ B)}{Prob\ (A)} \quad (3)$$

For that, naïve Bayes is particularly effective for large datasets and is simple to implement.

b. Support vector machine (SVM)

SVM is a supervised learning algorithm recognized for its high performance in regression and classification tasks. It is particularly effective in addressing quadratic programming problems involving linear equality and inequality constraints. SVM differentiates between groups by finding a hyperplane that maximizes the margin between classes, making it well-suited for tasks requiring precise boundary delineation. Although SVM may be slower than some classification techniques, its accuracy lies in its ability to model complex, non-linear, multidimensional boundaries effectively. Additionally, SVM is resilient to overfitting, even when models are complex with many parameters relative to the number of observations. These attributes make SVM an excellent choice for tasks such as text categorization, voice recognition, and handwritten text detection.

3.3.3. Phishing email classification based on an email's attachments

Email attachments are stored in an encrypted format, and the wide variety of attachment types and formats adds to the complexity of classifying them as benign or malicious. Malicious emails often use attachments as the primary method to deliver harmful payloads to recipients. These attachments may either be singularly malicious or attempt to blend in with other benign file types. Various algorithms are employed for classifying email attachments depending on their type.

A. Image classifier: Sequential neural network algorithm is used for image classification.

– Neural network (NN)

Input: The algorithm processes email attachment images resized to 256×256 pixels.

Output: The algorithm predicts the probability that an input image represents a malicious email attachment.

Overfitting and Generalization: Overfitting poses a significant challenge during neural network training. If the model is trained for too many epochs, it can overlearn patterns specific to the training data, resulting in excellent performance on the training set but poor generalization to new, unseen data. In such cases, the model fails to perform effectively on the test set. To minimize overfitting and improve generalization:

a. Optimal epoch selection: The model is trained for a carefully determined number of epochs to balance learning and generalization.
b. Validation data: A portion of the training data is reserved for validation to evaluate the model's performance after each epoch.
c. Metric monitoring: Loss and accuracy are tracked on both the training and validation datasets to identify when overfitting begins.

B. URL classifier: For URLs, the random forest algorithm (RF) is utilized.

– Random forest (RF)

RF is a popular ensemble-based approach applied to both classification and regression tasks. It operates by generating multiple decision trees during the training process. For classification problems, the final result is decided based on the majority vote among the trees, while for regression tasks, the prediction is calculated as the average of the outputs produced by all trees.

C. Document classifier: For document, the Naïve Bayes algorithm (NB) is utilized.

– Naïve Bayes (NB)

For documents, the process begins with feature selection using information gain (IG). IG calculates the information gain for each candidate feature, defined as the difference between two conditional probabilities. Tokens with a greater difference have a higher information gain, making them more relevant for classification. After feature selection, the NB algorithm is applied. Based on Bayes' Theorem, the NB classifier uses conditional probabilities to predict outcomes. It is straightforward and efficient, with its performance highly dependent on the quality and characteristics of the dataset.

4. EVALUATION AND RESULTS

This paper focuses on identifying spam emails and distinguishing them from legitimate ones by applying a neural network algorithm and four other machine learning algorithms. Two primary datasets are used for this study: the Enron Email Dataset, which includes hundreds of thousands of emails, and the SpamAssassin dataset, which contains 1,897 spam emails and 4,150 ham emails. Figure 5 and Figure 6 show the visualization of the number of spam and ham emails in the two combined datasets, with spam labeled as 1 and ham labeled as 0.

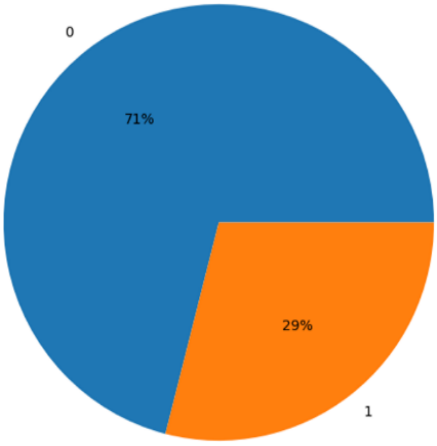


Figure 5. Enron email dataset

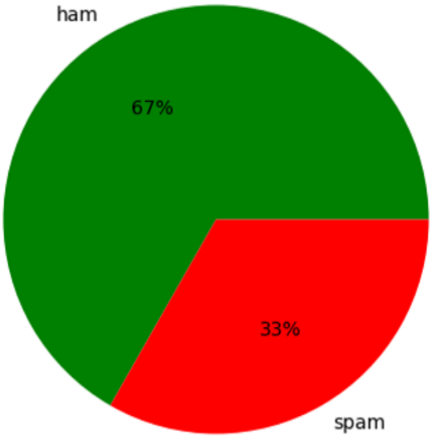


Figure 6. SpamAssassin dataset

In this study, publicly available datasets were utilized to evaluate the proposed phishing email classification model. Since commonly used email datasets such as Enron and SpamAssassin do not include file attachments, attachment samples were incorporated from an external attachment dataset to simulate phishing emails containing malicious files. This combination was performed to enable attachment-aware analysis, while acknowledging that the email body content and attachments may not always be semantically related. The performance of the proposed PECM is evaluated using confusion matrix, this matrix is commonly employed in prior research studies on spam email detection models. Table 7 illustrates the confusion matrix, by presenting the following indicators: true positives, true negatives, false positives, and false negatives. These indicators are utilized to assess the classification performance of the proposed model as follows:

a. False positive (FP) is the number of legitimate emails incorrectly classified as phishing
b. True negative (TN) is the number of legitimate emails classified as legitimate
c. False negative (FN) is the number of phishing emails incorrectly classified as legitimate
d. True positive (TP) is the number of phishing emails classified as phishing emails.

Table 7. The phishing email classification confusion matrix

Actual	Predict	
	0 - Legitimate	1 - Phishing
0 - Legitimate	TN	FP
1 - Phishing	FN	TP

The classifier's evaluation is performed using the accuracy, precision, and recall, as defined by (4) to (6).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (4)$$

$$Precision = \frac{TP}{(Tp+FP)} \quad (5)$$

$$Recall = \frac{TP}{(Tp+FN)} \quad (6)$$

Table 8 presents a comparison of performance metrics, including accuracy evaluated on the testing emails using various classifiers trained on the email dataset. The results in Table 8 indicate that applying the SVM to the email body yields higher accuracy compared to the NB classifier. Furthermore, for both the email header and attachments, all classifiers demonstrate strong accuracy, outperforming results reported in other studies. In this experiment, the performance of each classifier of the proposed PECM in each part of the email is illustrated in Figure 7.

a. Email header performance

The random forest (RF) classifier achieve accuracy=98.4%, precision=0.982 and recall=0.971. Which mean that it can handle the problem of phishing email that can be found in the header of the email with a best accuracy.

b. Email body performance

In the body of the email, the proposed PECM apply two classifier NB and SVM. The two classifiers achieve best accuracy but SVM outperform with accuracy=99%, precision=0.978 and recall=0.99.

c. Email attachment performance

The attachment of the email is categorized to three parts. The first part is the attached URL. The proposed PECM apply RF classifier which achieve a best accuracy=96.8%, precision=0.989 and recall= 0.994. The next part is the attached image, the proposed PECM apply the neural netNN) classifier to achieve accuracy=98%, precision=0.977 and recall=0.985. The last part is the document, the applied classifier is NB which achieve accuracy=99%, precision=0.983 and recall=0.98.

A comparison between the proposed PECM and related research intended to identify email spam is shown in Figure 8 and Table 9. Every spam detection technology used in other studies was judged to be successful. While some researchers aimed to increase accuracy through various methods, the proposed PECM achieved significant progress, providing either better accuracy or similar scores in most cases. It should be noted that the results of the other studies were taken directly from their respective publications and therefore may have been obtained using different datasets. As such, the comparison is indicative rather than definitive. Future work will focus on reproducing the compared methods using the same dataset and experimental conditions to ensure a fair and unbiased evaluation. Also, the inclusion of the semantic consistency check provided additional robustness in detecting attachment-based phishing emails, particularly in cases where deceptive filenames were used to increase user trust.

To provide a fair comparison with modern approaches, the performance of the proposed model is evaluated against a BERT-based model using reported results from recent literature. Transformer-based models such as BERT have shown strong performance in phishing email detection tasks due to their ability to capture contextual information [25].

As illustrated in Table 9, the proposed model outperforms the BERT-based approach reported in the literature in terms of accuracy. While BERT models benefit from deep contextual understanding, they often require higher computational resources and longer training times. In contrast, the proposed model achieves competitive performance with lower computational complexity, making it more suitable for real-time phishing detection scenarios.

Table 8. The accuracy metrics values of propose PECM

	Header	Body	Attachments		
			URL	Image	Documents
Algorithm	RF	NB	SVM	RF	NN
Accuracy	0.984	0.975	0.990	0.968	0.980
Precision	0.982	0.978	0.986	0.989	0.977
Recall	0.971	0.961	0.990	0.994	0.985

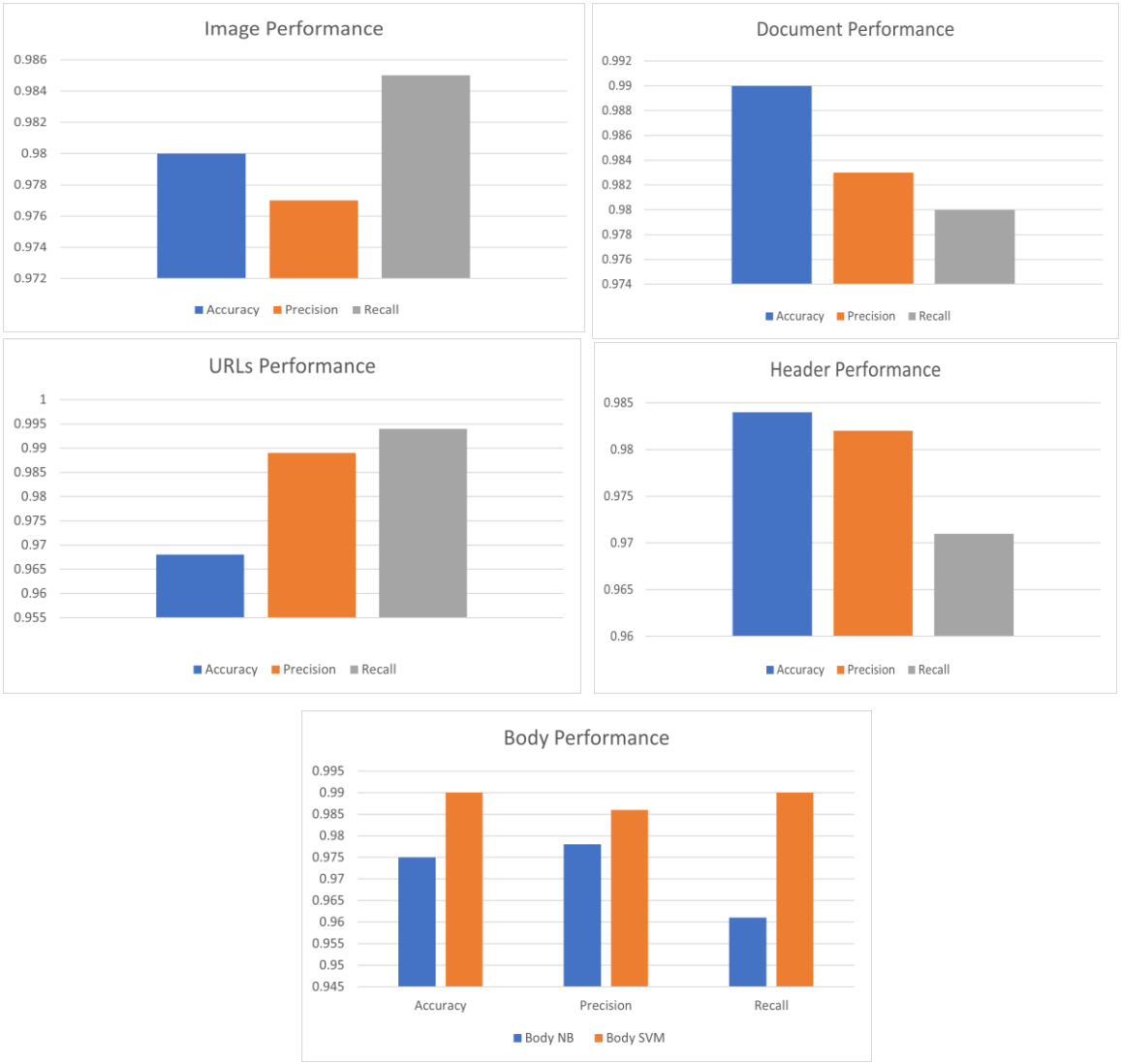


Figure 7. PECM performance achieved

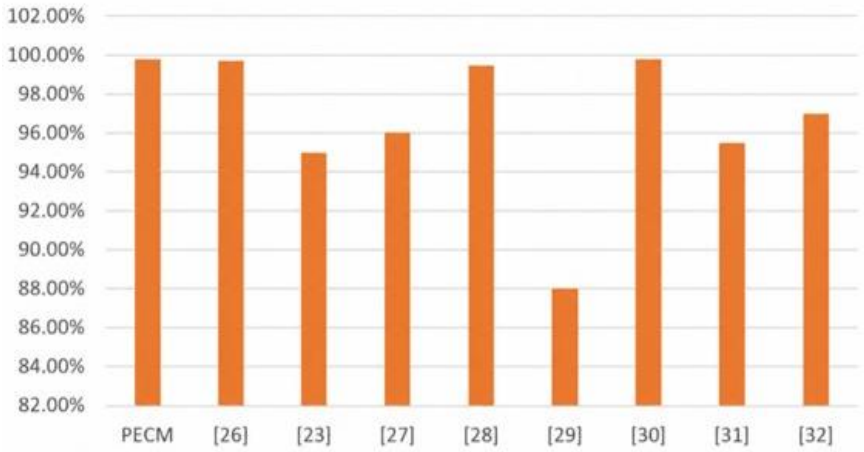


Figure 8. Comparison between PECM and other studies

Table 9. Comparison between propose PECM With relevant work

Ref	Dataset	Classifier	Accuracy
PECM	EnronSpam and SpamAssassin	Multi-modal ensemble (BoW/TF-IDF + classical ML)	Header - 98.4% Body - 97.5% URLs - 96.8% Image - 98% Docs - 99%
[26]	SpamAssassin and Nazario.	RF	99.7%
[23]	The dataset is proprietary and contains 2 million emails.	LSTM	95%
[27]	Initially, the emails were untagged, but they were later tagged using a clustering algorithm. PU corpora and EnronSpam.	SVM with RBF kernel	96%
[28]	SpamAssassin and PhishingCorpus.	RF	99.5%
[29]	Anomaly detection challenges from Kaggle and cybersecurity data mining (DMC 2010).	SVM	~88%
[30]	EnronSpam, SpamAssassin.	NN	EnronSpam: 98.33% SpamAssassin: 99.8%
[31]	SpamBase.	DT, RF	95.5%
[32]	PU corpora, EnronSpam.	Stacked Autoencoders	97%
[33]	Enterprise email dataset	BERT + CNN	97.5%
[34]	Public email datasets	1D-CNN + LSTM variants	90%

4.1. False negative analysis

Despite the high overall detection accuracy, a detailed examination of misclassified instances reveals that false negatives primarily occurred in emails containing minimal textual phishing indicators and benign-looking attachments. In several cases, the email body included neutral or generic language with limited suspicious keywords, reducing discriminative textual signals. Additionally, some attachments exhibited characteristics similar to legitimate documents, which may have reduced the attachment classifier's sensitivity. These observations highlight the importance of enhancing semantic consistency modeling in future work.

5. LIMITATIONS OF THE STUDY

First limitation of this study is that the synthetic dataset was constructed by merging attachments from the MEADE dataset with Enron messages, which does not fully preserve semantic coherence between email bodies and attachments. While this may not reflect real-world phishing scenarios where attachments are contextually linked to the email body, the dataset allows for a preliminary multi-modal evaluation of classifiers using both textual and attachment features. Future work could focus on creating datasets with stronger semantic alignment to better model realistic phishing attacks.

Second, the study relies on traditional machine learning algorithms for classification, without direct comparison to transformer-based models, which represent the current state-of-the-art in natural language processing. Additionally, the computational cost and latency of analyzing multiple email components simultaneously were not explicitly measured and will be considered in future work.

Although the proposed model achieved high accuracy (≈99%), this performance should be interpreted in light of the synthetic construction of the dataset. The attachment-bearing emails were generated by combining publicly available datasets, which may reduce semantic complexity compared to real-world phishing campaigns. Consequently, the classification task may be less ambiguous than naturally occurring phishing scenarios. Future research should validate the model using semantically consistent, real-world datasets to further assess generalization performance.

6. CONCLUSION AND FUTURE WORK

The primary objective of this study was to propose the PECM framework, designed to enhance the accuracy and efficiency of phishing email classification and detection using machine learning techniques. The framework integrates multiple email components, including body, header, and attachments, and leverages several machine learning algorithms for improved spam and phishing detection. Experimental results demonstrate that combining heterogeneous feature sources enhances detection robustness, particularly in complex phishing scenarios involving malicious attachments. While the previous accuracy metrics highlighted the effectiveness of individual classifiers, the present conclusion focuses on the implications of these results: the multi-modal approach enables more resilient detection of sophisticated phishing attempts that may evade traditional single-component models. Running three classifiers in parallel introduces additional computational overhead. The latency associated with analyzing multiple components per email

may affect real-time deployment in large-scale systems; however, the framework remains feasible for enterprise-level environments where security accuracy is prioritized. The literature review provided a taxonomy of previous studies in spam detection, malicious link detection, phishing email identification, and harmful attachments. It highlighted gaps in existing solutions, particularly the need for models that generalize beyond predefined features, which is crucial for mitigating zero-day phishing attacks. While the current PECM implementation relies on predefined features extracted via BoW and TF-IDF, future work will focus on adaptive feature extraction and online learning methods to enhance resilience against emerging threats.

Additionally, future research will incorporate semantic consistency analysis between email body and attachment content to address limitations of synthetic or mismatched datasets. The PECM framework will also be extended to detect non-English phishing emails and evaluate performance on previously unseen phishing campaigns, ensuring broader applicability and robustness in dynamic cybersecurity environments.

FUNDING INFORMATION

The authors received no funding for this work.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Lamees Mohamed Faraj		✓	✓		✓	✓	✓		✓		✓			✓
Sayed Abdel-Gaber	✓			✓		✓		✓		✓		✓	✓	
Hanan Fahmy	✓	✓		✓			✓	✓		✓		✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

CODE AND DATA AVAILABILITY

The data and code are available at this link: https://github.com/lamees-mohamed/PECM-Phishing-Detection/upload/main.

REFERENCES

[1] L. Halgaš, I. Agrafiotis, and J. R. C. Nurse, "Catching the phish: Detecting phishing attacks using recurrent neural networks (RNNs)," in *Information Security Applications. WISA 2019. Lecture Notes in Computer Science*, Vol 11897., Springer, Cham, 2020, pp. 219–233.

[2] S. Gibson, B. Issac, L. Zhang, and S. M. Jacob, "Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms," *IEEE Access*, vol. 8, pp. 187914–187932, 2020, doi: 10.1109/ACCESS.2020.3030751.

[3] Verizon Business, "2024 data breach investigations report," *verizon.com*, 2024. https://www.verizon.com/business/resources/reports/dbir/ (accessed Nov. 30, 2025).

[4] Q. Cui, "Detection and analysis of phishing attacks," Thesis, University of Ottawa, 2019.

[5] S. K. Birthriya and A. K. Jain, "A comprehensive survey of phishing email detection and protection techniques," *Information Security Journal: A Global Perspective*, vol. 31, no. 4, pp. 411–440, 2021, doi: 10.1080/19393555.2021.1959678.

[6] T. Muralidharan and N. Nissim, "Improving malicious email detection through novel designated deep-learning architectures utilizing entire email," *Neural Networks*, vol. 157, pp. 257–279, Jan. 2023, doi: 10.1016/j.neunet.2022.09.002.

[7] A. Alhogail and A. Alsabih, "Applying machine learning and natural language processing to detect phishing email," *Computers & Security*, vol. 110, p. 102414, Nov. 2021, doi: 10.1016/j.cose.2021.102414.

[8] K.-N. Tran, M. Alazab, and R. Broadhurst, "Towards a feature rich model for predicting spam emails containing malicious attachments and URLs," in *Eleventh Australasian Data Mining Conference (AusDM 2013)*, 2013, pp. 1–10.

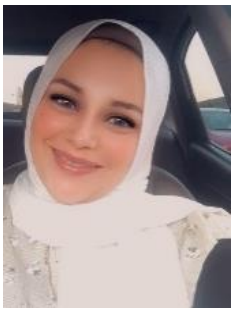
[9] N. Govil, K. Agarwal, A. Bansal, and A. Varshney, "A machine learning based spam detection mechanism," in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2020, pp. 954–957, doi: 10.1109/ICCMC48092.2020.ICCMC-000177.

[10] G. Mohanan, D. M. Padmanabhan, and G. S. Anisha, "Classifying emails into spam or ham using ML algorithms," in *Data Science and Security. Lecture Notes in Networks and Systems*, Vol 290., Springer, Singapore, 2021, pp. 214–221.
[11] B. Sonare, G. J. Dharmale, A. Renapure, H. Khandelwal, and S. Narharshettiwar, "E-mail spam detection using machine learning," in *2023 4th International Conference for Emerging Technology (INCET)*, May 2023, pp. 1–5, doi: 10.1109/INCET57972.2023.10170187.
[12] C. Beaman and H. Isah, "Anomaly detection in emails using machine learning and header information," *arXiv preprint arXiv:2203.10408*, pp. 1–10, 2022, doi: 10.48550/arXiv.2203.10408.
[13] I. AbdulNabi and Q. Yaseen, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021, doi: 10.1016/j.procs.2021.03.107.
[14] A. Sharaff and U. Srinivasarao, "Towards classification of email through selection of informative features," in *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, Jan. 2020, pp. 316–320, doi: 10.1109/ICPC2T48082.2020.9071488.
[15] T. Peng, I. Harris, and Y. Sawa, "Detecting phishing attacks using natural language processing and machine learning," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, Jan. 2018, pp. 300–301, doi: 10.1109/ICSC.2018.00056.
[16] V. Listík, J. Šedivý, and V. Hlaváč, "Email image spam classification based on ResNet convolutional neural network," in *Proceedings of the 6th International Conference on Information Systems Security and Privacy*, 2020, pp. 457–464, doi: 10.5220/0008956704570464.
[17] A. P. Singh, "Image spam classification using deep learning," San Jose State University, San Jose, CA, USA, 2018.
[18] T. Kumaresan, S. Sanjushree, K. Suhasini, and C. Palanisamy, "Image spam filtering using support vector machine and particle swarm optimization," *National Conference on Information Processing and Remote Computing*, vol. NCIPRC2015, no. 1, pp. 17–21, 2015.
[19] M. Soranamageswari and C. Meena, "Statistical feature extraction for classification of image spam using artificial neural networks," in *2010 Second International Conference on Machine Learning and Computing*, 2010, pp. 101–105, doi: 10.1109/ICMLC.2010.72.
[20] Y. Wei, M. Nakayama, and Y. Sekiya, "Enhancing generalization in phishing URL detection via a fine-tuned BERT-based multimodal approach," *IEEE Access*, vol. 13, pp. 131197–131216, 2025, doi: 10.1109/ACCESS.2025.3591843.
[21] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," *Decision Support Systems*, vol. 107, pp. 88–102, Mar. 2018, doi: 10.1016/j.dss.2018.01.001.
[22] N. Ameeno, K. Sherry, and K. Gagneja, "Using machine learning to detect the file compression or encryption," *Amity Journal of Computational Sciences (AJCS),* vol. 3, no. 1, pp. 31–36, 2019.
[23] Q. Li, M. Cheng, J. Wang, and B. Sun, "LSTM based phishing detection for big email data," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 278–288, Feb. 2022, doi: 10.1109/TBDATA.2020.2978915.
[24] D. Gaurav, S. M. Tiwari, A. Goyal, N. Gandhi, and A. Abraham, "Machine intelligence-based algorithms for spam filtering on document labeling," *Soft Computing*, vol. 24, no. 13, pp. 9625–9638, Jul. 2020, doi: 10.1007/s00500-019-04473-7.
[25] P. Pandey, C. Agrawal, and T. N. Ansari, "A hybrid algorithm for malicious spam detection in email through machine learning," *International Journal of Applied Engineering Research*, vol. 13, no. 24, pp. 16971–16979, 2018.
[26] A. A. Akinyelu and A. O. Adewumi, "Classification of phishing email using random forest machine learning technique," *Journal of Applied Mathematics*, vol. 2014, pp. 1–6, 2014, doi: 10.1155/2014/425731.
[27] Y. Tan, Q. Wang, and G. Mi, "Ensemble decision for spam detection using term space partition approach," *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 297–309, Jan. 2020, doi: 10.1109/TCYB.2018.2868794.
[28] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proceedings of the 16th international conference on World Wide Web*, May 2007, pp. 649–656, doi: 10.1145/1242572.1242660.
[29] S. A. Khamis, C. F. M. Foozy, M. F. A. Aziz, and N. Rahim, "Header based email spam detection framework using support vector machine (SVM) technique," in *Recent Advances on Soft Computing and Data Mining. SCDM 2020. Advances in Intelligent Systems and Computing*, Vol 978., Springer, Cham, 2020, pp. 57–65.
[30] A. Barushka and P. Hájek, "Spam filtering using regularized neural networks with rectified linear units," in *AI*IA 2016 Advances in Artificial Intelligence. AI*IA 2016. Lecture Notes in Computer Science*, Vol 10037., Springer, Cham, 2016, pp. 65–75.
[31] A. Sharaff and H. Gupta, "Extra-tree classifier with metaheuristics approach for email classification," in *Advances in Computer Communication and Computational Sciences. Advances in Intelligent Systems and Computing*, Vol 924., Springer, Singapore, 2019, pp. 189–197.
[32] G. Mi, Y. Gao, and Y. Tan, "Apply stacked auto-encoder to spam detection," *Advances in Swarm and Computational Intelligence*, *ICSI 2015*, Springer, Cham, vol. 9141, 2015, doi: 10.1007/978-3-319-20472-7_1.
[33] B. B. Gupta, A. Gaurav, V. Arya, R. W. Attar, and S. Bansal, "Advanced BERT and CNN-based computational model for phishing," 2024, doi: 10.32604/cmes.2024.056473.
[34] N. Altwaijry, I. Al-Turaiki, R. Alotaibi, and F. Alakeel, "Advancing phishing email detection: A comparative study of deep learning models," *Sensors*, vol. 24, no. 7, p. 2077, Mar. 2024, doi: 10.3390/s24072077.

BIOGRAPHIES OF AUTHORS

Lamees Mohamed Faraj received the B.Sc. degree in computer science from Helwan University, Egypt. She is currently pursuing her M.Sc. degree in computer science. Her research interests include cybersecurity, phishing email detection, machine learning, artificial intelligence, and natural language processing. She can be contacted at email: Lamees_m_frj@cic-cairo.com.

Sayed Abdel-Gaber received the Ph.D. degree in information systems. He is currently the Dean of the Canadian International College, Egypt. He is also the Digital Transformation Manager at Helwan University, Egypt. His research interests include data science, medical informatics, and software engineering. He can be contacted at email: sgaber14@gmail.com



Hanan Fahmy received the Ph.D. degree in information systems from the Faculty of Artificial Intelligence, Helwan University, Egypt. She is currently an associate professor with the Department of Information Systems, Faculty of Artificial Intelligence, Capital University, Egypt. Her research interests include cybersecurity, machine learning, artificial intelligence, big data, data analytics, IoT data analytics, advanced database management, and software engineering. She can be contacted at email: hann.fahmy@fci.capu.edu.eg.