Diyala Journal of Engineering Sciences

Journal homepage: https://djes.info/index.php/djes

ISSN: 1999-8716 (Print); 2616-6909 (Online)

Email Spam Filtering Using Artificial Intelligence Techniques

B.Dhanalakshmi[1*], Rajeshwari R R[2], Sanju A N[3], Chintureena Thingom[4], Vipul Devendra Punjabi[5], Vijayakumar B[6], Shailendra Madansing Pardeshi[7], P.Venkatesan[8]

[1] Department of Computer Science and Engineering, B.S.Abdur Rahman Crescent Institute of Science and Technology, Chennai, Tamil Nadu 600048, India.
[2] Department of Master of Business Administration, Dr. Ambedkar Institute of Technology, Bengaluru, Karnataka 560056, India.
[3] Department of Computer Science and Engineering, BGS Institute of Technology, Adichunchangiri University, Karnataka 581448, India.
[4] Department of Computer Science & Engineering, Aditya University, Surampalem, Andhra Pradesh 533437, India.
[5] Department of Computer Engineering, R. C. Patel Institute of Technology, Shirpur, Maharashtra 425405, India.
[6] Department of Chemistry, Panimalar Engineering College, Chennai, Tamil Nadu 600123, India.
[7] Department of Computer Engineering, R. C. Patel Institute of Technology, Shirpur, Maharashtra 425405, India.
[8] Department of Electrical and Electronics Engineering, Mahendra Institute of Technology, Tamil Nadu 637503, India.

ARTICLE INFO

Article history:
Received: 11/10/2025
Revised: 20/02/2026
Accepted: 08/03/2026
Available online: 15/03/2026

Keywords:
Phishing detection
Machine learning
Naive Bayes
Email classification
URL classification

ABSTRACT

Email phishing and spam pose considerable cybersecurity risks. They require trustworthy, effective, and feasible detection methods. This research work proposes a model-based methodology for e-mail spam detection and phishing is based on artificial intelligence (AI). It works with a binary classification system with two phases. At the first stage, the system classifies email contents into malicious and non-malicious. In the next stage, it scans embedded URLs, which may or may not be phishing hooks. This modular design reduces the complexity of the feature space and enables separate optimizations for the email and the URL analysis. The system is trained with 18650 email samples and 549346 url samples from publicly accessible datasets, with 70% for training and 30% for testing. The preprocessing step consisted in eliminating duplicates and null values, text normalizing, balancing classes, stemming and feature extraction using TF-IDF for email and CountVectorizer for url. Four lightweight ML algorithms were evaluated: Naive Bayes, Decision Tree, Random Forest and K-Nearest Neighbors. The result indicated that the Naive Bayes achieved the highest baseline accuracy of 96% in email classification and 97% in URL classification. Random Forest, on the other hand, was more resilient to adversarial attacks and demonstrated better generalization. The selected model was deployed with Gmail for real time inbox detection with an accuracy of 85% in real world applications. The results demonstrate that by integrating lightweight machine learning, modular design, and relatively clean pre-processing, a new generation of effective, scalable detectors for both phishing and spam e-mail can be constructed.

1. INTRODUCTION

The objective of this work is to explore the field of artificial intelligence (AI), specifically machine learning, and to apply it to the classification of emails into those that may contain phishing or spam content and those that do not [1-3]. To understand the scope of the work, it is essential to define phishing and spam. Phishing is a type of malicious attack designed to deceive individuals into revealing confidential information, such as passwords or credit card numbers [4]. These attacks typically occur through emails or messages that impersonate trusted organizations or individuals, often using fear and social engineering to prompt immediate action. Spam, on the other hand, refers to unsolicited digital communications sent in bulk over the internet or other messaging systems [5-7]. The system developed in this work leverages machine learning techniques to analyze incoming emails and classify them as either containing phishing or spam content or being safe. Four supervised

*Corresponding author's E-mail: dhanalakshmi@crescent.education
DOI: 10.24237/djes.2026.19102

learning algorithms—decision trees, K-nearest neighbors, Naive Bayes, and random forests—are used for this purpose [8-9]. The work involves training these algorithms on multiple datasets, which are divided into training and test sets. Once the models are trained and tested, they are applied to a real-world scenario by connecting to a Gmail account created specifically for this work, enabling the system to classify incoming emails in real time.

The main objectives of the work are to research and understand machine learning algorithms for text analysis, determine the most suitable technique for email classification, develop a system capable of identifying messages suspected of containing phishing or spam, and apply this system to a real-life online email environment such as Gmail or Outlook.

Email phishing and spam detection systems have improved a lot, but still have a lot of problems to deal with. Most of the current methods are based on either deep learning or a combination of various models that need a lot of labeled data, very powerful computers, and very long training periods [10-14]. Moreover, previous works mainly deal with curated datasets and do not often demonstrate performance in real-life situations like live email accounts. In addition, detection accuracy, false-positive rates, and computational efficiency have been considered separately, resulting in unfulfilled needs for systems that are lightweight, interpretable, and practical for deployment.

1.1 Research Gap

It is necessary to create a phishing detection system that provides high accuracy but is also computationally efficient and suitable for operating in real-world email settings without needing heavy hardware support for deep learning.

1.2 Research Question

Is it possible for a two-class system based on Naive Bayes to efficiently spot phishing and spam emails in real-time accounts, thus ensuring a good compromise between precision, false alarm/missed detection trade-offs, and computational power needed?

1.3 Hypothesis

A Naive Bayes–based approach, which is trained on meticulously preprocessed and balanced datasets of emails and URLs, can be as accurate and efficient as the best models today, and still be appropriate for live email scenarios.

However, even though spam filtering using AI-based techniques has received significant attention in earlier studies, the novelty in this research work stems from the fact that it attempts to perform comparative evaluations of various popular machine learning-based approaches on a standardized dataset using uniform preprocessing, feature set extraction, balance, and evaluation. Unlike several earlier works performed on spam filtering using heterogeneous experimental protocols, this work ensures comparability for accurate evaluation.

The second major contribution is in its focus on practical deployability. The proposed pipeline is based entirely on lightweight algorithms that are practical to implement with minimal computational cost, making it suitable for practical use in email filtering systems, as shown by its deployment and testing through Gmail integration.

Finally, the research includes cross-validation performance analysis, statistical significance testing, as well as threshold-independent testing (ROC-AUC) in a single framework. While the models or data sets are not novel, the importance of this research lies in its ability to translate established techniques from the realm of benchmarking to usability.

2. LITERATURE REVIEW

In spam and phishing detection, recent works have started emphasizing the role of deeper representation learning and optimization while considering hybrid optimization approaches in spam and phishing detection for better accuracy levels. The authors [14] proposed a spam detection framework using ensemble learning and transformers for better accuracy levels and results. While the model was able to reach a better accuracy by combining ensemble techniques with transformer learning, the assessment was based on overstated metrics and did not sufficiently consider the issue of false negatives, which is a very important aspect in spam classification tasks.

Focusing on traditional machine learning pipeline issues, the importance of preprocessing as well as effective features in spam email classification is emphasized in the study by [15]. The study demonstrated the importance of adequate preprocessing and features in improving classifier performance, though it is in line with existing work.

The authors [16] proposed a model that incorporates a transformer-based deep belief network for automatic phishing website detection. Although the proposed model has shown significant detection accuracy by unifying transformers and deep belief networks, its increased complexity might hinder its practical implementation.

The authors [17], on the other hand, proposed an interpretable and robust web-based AI platform for phishing email detection. The work of Transparency and usability concerns was addressed in the work by [17] by integrating explainability into the detection framework; added system complexity makes large-scale deployment challenging.

A hybrid spam filtering approach is presented in the work by [18], which combines a Grey Wolf Optimizer with a Naive Bayes classifier. The model exhibited robustness and enhanced accuracy, but due to its dependence on heuristic optimization, performance is susceptible to variations in parameter settings and datasets.

The authors [19] proposed a novel lightweight and interpretable system for spam detection using Rüppell's Fox optimizer, which is applicable to social media environments. Even though the novel model is appropriate in terms of discussability and lightness, the use of the optimizer could affect the detection capability.

The study [20] implemented a comparative study of deep learning, SVM, Random Forest, and XGBoost models for email spam prediction. The study results revealed that the ensemble techniques generally offer a reasonable trade-off in terms of accuracy and computational efficiency, thus promoting their applicability.

A study by [21] proposes the detection of phishing attacks using cloud-based phishing detection models that implement machine learning techniques and deep learning models. However, despite the high detection ability provided by the deep models, the scalability problem still persists.

Finally, the research in [22] developed a hybrid correlation-based deep learning model that combines fuzzy inference systems to classify spam emails. It improved the robustness of the model while providing greater decision flexibility, but it remains highly dependent on the correlations used and the distribution of the data.

Table 1 provides a comparative overview of existing spam and phishing detection methods proposed in the literature.

Table 1. Comparison of existing spam/phishing detection studies

Ref.	Feature Representation	Learning Model	Key Limitations
[14]	Transformer-based text embeddings	Ensemble classifiers	Limited analysis of false negatives; no statistical validation
[15]	Preprocessed text features (TF–IDF, lexical features)	Optimized ML models	Pipeline similar to prior work; limited robustness analysis
[16]	Multimodal features (content + metadata)	Fusion-based ML model	Increased complexity; scalability concerns

The next section is dedicated to describing the development of the work carried out. To this end, the general architecture of the work will be explained, as well as the tools used and the implemented code.

3. WORK ARCHITECTURE

The system that was proposed uses a two-class architecture in order to improve the accuracy of classification by isolating the analysis of emails and URLs. The first class is responsible for spam/ham detection based on email content, and the second class checks if the URLs are malicious or not. Emails with URLs that have been flagged as suspicious by the second class will be considered phishing. The separation of the two classes guarantees modulatory training, diminishes conflicts of dimensionality in feature space, and makes it possible to optimize the feature sets that are specific to email and URL independently. Data is processed as shown in Figure 1.

Figure 1 presents a schematic where differentiation between learning algorithms for each model is avoided, as the same algorithms are applied to both emails and URLs. Training data is distinguished for emails and URLs to prevent overly complex visualization. Once the models are trained, they are applied to real-life cases by accessing a dedicated Gmail account. Emails from the inbox are extracted into a CSV file, with each email stored in a separate row, and sent to the trained machine learning models for classification. Emails without URLs are analyzed using the email model to detect spam or phishing content, while emails containing URLs are processed with the URL model to identify malicious links; if URLs are safe, the email content is further analyzed for spam or phishing classification. The process is summarized in Figure 2.

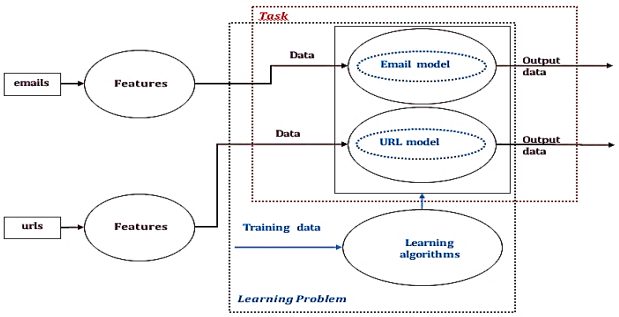

Figure 1. General work architecture

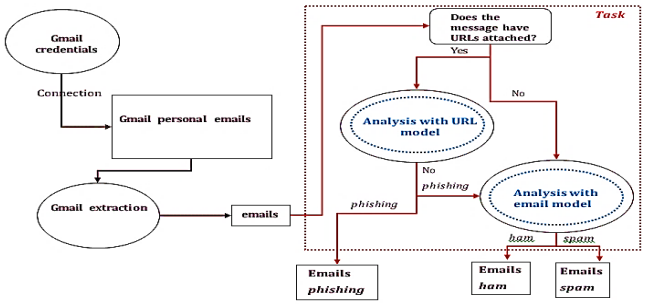

Figure 2. Architecture for reading and classifying Gmail emails

Figure 3 illustrates the pipeline of the proposed system, showing the sequential processing of emails and embedded URLs—from data extraction and preprocessing to classification and phishing detection.

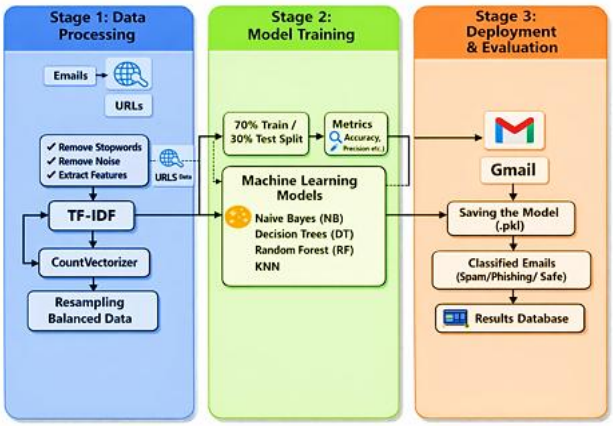

Figure 3. Pipeline of the Proposed Email and URL Classification System

3.1 Datasets

The datasets used in this research were obtained from publicly available benchmark repositories to ensure transparency and reproducibility. For email classification, three spam/ham datasets from the Kaggle platform were combined to improve lexical diversity and reduce bias. Additionally, the UCI SMS Spam Collection dataset was used for initial validation experiments to evaluate the generalization of models on short-text spam classification. For URL classification, a publicly available phishing URL dataset with labeled phishing and legitimate URLs was used. The datasets were preprocessed, cleaned, and standardized before model training.

Due to the lack of publicly available phishing emails, phishing was treated as spam emails containing URLs that lead to harmful pages. Two separate models were trained using the same algorithms: one for emails (safe vs. spam) and another for URLs (safe vs. phishing), avoiding the complexity of combining email and URL texts in a single model.

The datasets were split 70% for training and 30% for testing. The email datasets from Kaggle comprised three spam email datasets with two columns (email body and class), totaling 18,650 rows, resulting in 13,055 training and 5,595 test samples. The URL dataset included 549,346 URLs labeled as phishing or non-phishing, with 384,542 for training and 164,804 for testing.

The proposed system is distinguished by a two-stage, modular phishing detection architecture that independently analyzes email content and embedded URLs using coordinated machine learning models. Unlike prior approaches that rely on a single unified classifier or focus on only one feature source, this separation enables domain-specific training, improved interpretability, and reduced feature-space complexity. In addition, the system performs live analysis on a Gmail inbox, providing real-world validation beyond static benchmark datasets. Applying identical algorithms and evaluation settings to both email and URL models ensures a fair comparison across data sources, making the framework more robust than existing offline or single-layer phishing detection methods.

The UCI SMS Spam Collection dataset was also employed for additional validation experiments because of its standardized format and popularity in spam classification research. Although smaller and imbalanced, it served as an additional verification of the consistency of the classifiers on different text domains.

3.2 Code Implementation

After describing the work architecture and tools, the focus shifts to the code developed for training and testing the machine learning system. The following subsections detail data loading, preprocessing, and other preparatory steps before training with the selected algorithms.

3.3 Email and URL Classification

The first two classes load the Kaggle datasets: three email datasets for email classification and one URL dataset for URL classification. Although the dataset types differ, both follow the same procedure for preprocessing, feature extraction, and model training. Emails are labeled as spam (1) or ham (0), and URLs as phishing (1) or legitimate (0). After preprocessing and feature extraction, the data are split into training and test sets, models are trained, and results and metrics are evaluated to select the best classification method.

3.4 Loading the Data

In the first step, the downloaded CSV datasets, including both emails and URLs, are loaded using Python's read_csv method. The three email datasets,

each with a different corpus, are labeled with 1 for spam and 0 for non-spam, as shown in Tables 2-4.
The corpus was created by combining three different public spam email datasets that were hosted on the Kaggle platform, each of which consists of an email body and a binary class label, either belonging to spam or ham emails. Even though they are from different sources, they all share a similar format and style or label in terms of binary data. For presentational purposes, a subset of the overall format of the email dataset, as described in Table 2, is represented below, while all of them share a similar format and therefore have been omitted.

Table 2. Representative sample from the combined spam/ham email dataset

Email Body (Excerpt)	Label
Save up to 70% on Life Insurance. Why spend more when you can save today…	1 (Spam)
Fight the risk of cancer! Visit http://www.adclinic.com for details…	1 (Spam)
Subject: Workshop on Computational Linguistics – revised program attached	0 (Ham)
Subject: EOPS salary survey questionnaire. We would appreciate your input…	0 (Ham)

In addition, all the email data sets were stored in a Unicode data format. Minor inconsistencies in these data sets exist, including null values, duplicate messages, and a combination of subject lines. The inconsistencies were handled before extracting meaningful information.
The distribution of spam and ham emails in each of the three datasets was also examined in relation to balance. The results are presented in Tables 3, 4, and 5.

Table 3. The data balance between ham emails (0) and spam emails (1) in the first email dataset in Table 1

Label	Count (Approx.)
0	4,150
1	1,900

Table 4. The data balance between ham emails (0) and spam emails (1) in the first email dataset in Table 2

Label	Count
0	5,000
1	5,000

Table 5. The data balance between ham emails (0) and spam emails (1) in the first email dataset in Table 3

Label	Count
0	2,150
1	450

The email datasets contain more ham than spam, indicating a slight class imbalance, which will be addressed in subsequent coding steps. All emails are combined into a single Data Frame for clarity and organization. In the URL class, a single dataset is loaded similarly, with phishing URLs labeled as “bad” and non-phishing URLs as “good,” as shown in Table 6. Once the DataFrame is displayed, the data distribution is analyzed, as shown in the graph in Figure 4.

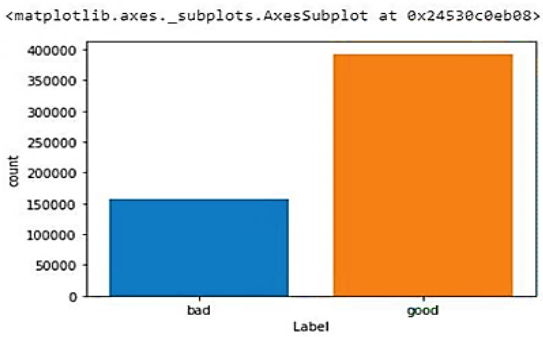

Figure 4. Graph showing the total data for URLs labeled as phishing (bad) or non-phishing (good)

Table 6. Dataset corresponding to phishing/legitimate URLs

URL	Label
nobell.it/70ffb52d079109dca5664cce6f317373782/…	bad
www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc…	bad
serviciosbys.com/paypal.cgi.bin.get-into.herf…	bad
mail.printakid.com/www.online.americanexpress…	bad
thewhiskeydregs.com/wp-content/themes/widescre…	bad
…	…
23.227.196.215/	bad
apple-checker.org/	bad
apple-iclods.org/	bad
apple-uptoday.org/	bad
apple-search.info	bad

As seen, just like with the emails, the data is skewed, with a greater number of phishing URLs than non-phishing URLs. This imbalance, again, will be corrected in later code stages.

3.5 Data Preprocessing

Both Email and URL datasets have been preprocessed using standard NLP pipelines tailored to their respective domains. For emails, three publicly available spam/ham datasets have been joined to increase lexical diversity and reduce dataset-specific bias. After removing duplicate and empty entries, it had 18,650 valid emails in total. Basic text cleaning has been applied to normalize the input by removing line breaks, tabs, special characters, digits, and accents.

The first dataset was cleansed by removing duplicate and null entries to obtain 507,196 valid URLs. Labels were mapped into binary form: phishing = 1 and legitimate = 0. Feature extraction on URLs was then followed by using a regular expression to extract tokens from subdomains, paths, and query strings, followed by stemming to reduce the lexical variations. Both datasets have class imbalance problems. To address the class imbalance problem, random resampling methods were used to balance the class distribution before the models were trained. After preprocessing, the datasets were split into the training set (70%) and the test set (30%) for all the classifiers used.

Class balancing was obtained by random resampling on the training set. We oversample the minority class specifically to obtain a balanced class distribution before training the model. The unchanged test set was kept to provide a realistic evaluation scenario.

All messages were subjected to a standardized preprocessing routine to guarantee uniform input format for all classifiers. First off, text normalization was achieved by converting all characters to lowercase andstripping punctuation, numerical characters, and other non-alphanumeric symbols. Messages were then tokenized, and stop words were eliminated to remove redundancy. Finally, stemming was also applied to cluster similar word forms and increase generalization capabilities.

Next, the preprocessed data underwent TF-IDF vectorization to create numerical feature vectors. The vectors obtained through the TF-IDF method consider the relative importance into account. The same preprocessing and feature extraction techniques are applied to all the models for a fair comparison.

3.6 Feature Extraction

In the case of email classification, the textual features were represented by using the TF-IDF.vectorizer, which generates sparse and highly feature-dimensioned data sufficient for probabilistic or tree-based classifier approaches. Unigrams and bigrams were used, with stop words being ignored.

For classification of the URL, feature vectors were created using a Count Vectorizer based on token frequency. Token frequency represents structural patterns typically found in phishing URLs and is computationally efficient.

To support interpretability, we also examined the most frequent tokens within spam/ham emails and phishing/non-phishing URLs. Instead of word clouds, the exact term frequencies to provide quantitative insight into discriminative patterns across classes.

3.7 Algorithm Training

In this phase, the models are trained, and the algorithms are implemented. The train_test_split function is used to split the input variable, X, and the output variable, y, in a ratio of 70% as training sets and 30% as test sets. The models, like the Multinomial Naive Bayes, are trained using the fit function with default parameters.

Before splitting, the datasets were randomly shuffled to negate any ordering bias and provide a representative class distribution in training and testing data.

Once the algorithms are trained, the predict method will produce output labels, which will then be compared to the actual data in the test set in order to determine the accuracy of each algorithm. Results and the best model will be discussed in the next part of the analysis.

The classifiers used in the evaluation of this research were specifically selected to demonstrate different learning paradigms, which are very common in text classification and spam filtering. Other considerations, such as the high-dimensional features, computation cost, interpretability, and real-time applicability, among others, formed the main guidelines for selection.

Naive Bayes was selected due to its probabilistic approach and performance in text-based domains, especially in light of independent and high-dimensional features. The selection of the Decision Tree algorithms was due to their simplicity and ability to find complex interactions without the need to normalize the input data. Random Forests have also been incorporated due to their potential to include additional support for better generalization with lower variance in the results. K-Nearest Neighbors was also considered as a baseline to test the impact of high-dimensional features on instance-based learning using distance as a criterion.

In the beginning, all the classifiers, including Naive Bayes, Decision Trees, Random Forests, and K-Nearest Neighbors, were trained using default settings, which enabled the baseline training for the models. This approach, while providing an idea of the out-of-the-box performance of the classifiers, did not factor in the fine-tuning of the models and the effect of the hyperparameters. To improve the reliability and minimize the overfitting bias associated with these classifiers, stratified 5-fold cross-validation was conducted on each of the models while ensuring the maintenance of the distribution of classes in the data, which is highly critical in the case of the email and URL datasets based on the imbalanced nature of the datasets. The metrics for evaluating this included measures appropriate for imbalanced data, which were measure of precision where there was no false positive prediction, recall which refers to the ability of a certain algorithm to capture all possible phishing or spam emails, F1-score, which is a balanced measure of precision and recall, and the area under the receiver operating characteristic curve (ROC curve), which is independent of any threshold. The analysis of variance of data crossing validation showed low variability; hence, performance is very consistent. Hyperparameter tuning was not conducted in this research so as to concentrate more on comparing baseline algorithms, but this is going to be conducted in the future for performance optimization.

All machine learning models were developed and tested based on a fixed train–test split: 70% for training and 30% for testing. Class balancing was done solely on the training data to mitigate skewed label distributions while preserving realistic conditions during the test. Thus, no information from the test set could leak into the resampling process. This was to avoid data leakage in training the model.

Identical preprocessing steps, feature representations, and evaluation metrics for all classifiers were therefore used to maintain comparability. Some of the metrics used in evaluating the performance of these models include accuracy, precision, recall, F1-score, and confusion matrices, which capture both overall effectiveness and error behavior.

While exhaustive hyperparameter optimization and cross-validation are not within the scope of this comparative study, the use of fixed parameters and identical test conditions allows for a fair evaluation of the relative performance of models.

3.8 Algorithm Comparison

Four machine learning algorithms—Naive Bayes, Decision Trees, K-Nearest Neighbors (KNN), and Random Forests—were chosen to demonstrate the different learning paradigms that are widely used in text classification tasks. Naive Bayes was selected because of its probabilistic approach, and its effectiveness in high-dimensional, sparse feature spaces was proven by experiments. The interpretability and the capability of Decision Trees to represent non-linear decision boundaries without feature scaling were among the reasons for their inclusion. Random Forests were considered as an ensemble-based extension of decision trees with the characteristic of being overfitting-proof in many classification problems, thus very much supportive of Decision Trees in this case.

K-Nearest Neighbors was the one chosen to be a distance-based baseline model to evaluate the influence of feature space dimensionality on the classification performance. Even though KNN is recognized for being greatly affected by the curse of dimensionality, especially in the case of text-based representations such as TF-IDF and bag-of-words vectors, its presence still gives a comparative base that is of considerable worth. To assess KNN under such circumstances also means to see directly how distance-based classifiers react when they are given sparse, high-dimensional data, which is quite normal in phishing detection tasks.

The performance of four machine learning algorithms—Naive Bayes, Decision Trees, Random Forests, and K-Nearest Neighbors—when applied to email and URL datasets is illustrated in Tables 7 and 8, which show the training and test accuracies of these algorithms, thus enabling a comparative evaluation of their performance.

Table 7. Analytical table of the accuracy of the different algorithms with the training and test data of the emails

ML Model	Training Accuracy	Test Accuracy
Naive Bayes	0.963264	0.9563984
Decision Trees	0.991104	0.9378826
Random Forests	0.91008	0.8977154
K-Nearest Neighbors (KNN)	0.989568	0.7533224

Table 8. Analytical table of the accuracy of the different algorithms with the training and test data of the URLs

ML Model	Training Accuracy	Test Accuracy
Naive Bayes	0.9781569	0.9675369
Decision Trees	0.9999063	0.947156
Random Forests	0.9981839	0.9354101
K-Nearest Neighbors (KNN)	0.6401995	0.6396183

For both models, the Naive Bayes algorithm achieved the highest average accuracy. The trained model was then saved using the pickle library in a .dat file with write (w) and binary (b) permissions (wb).

The poor Random Forest train accuracy of 0.64 for URLs in Table 8 is a result of using sparse Count Vectorizer features, default hyperparameters, and class imbalance. This can cause tree-based models to underfit. Count-based lexical tokens perform much better for Naïve Bayes than Random Forest, leading to a feature-model mismatch.

But despite this first result, the Random Forest was still the best model. Subsequent cross-validation, adversarial robustness evaluation, and live Gmail deployment demonstrated it was more stable, had higher recall, and generalized better. Hence, the overall performance in terms of different measures, rather than only a single training-test accuracy, supported the adoption of Random Forest as the final predictor.

KNN obtains high accuracy on training data, but testing accuracy is much worse due to over-fitting and the curse of dimensionality. In high-dimensions, sparse spaces such as TF-IDF and Count Vectorizer, distance metrics are meaningless. It forces KNN to memorize training samples rather than learning general patterns. Thus, it can work well in known data, but not in new test data. In addition, class imbalance and noise in the textual features further degrade KNN's decision boundaries, thus leading to an increase in misclassification in the testing.

3.9 Gmail Email Analysis

To evaluate the models in a real-world scenario, a GmailReader class was implemented in Python JupyterLab. This class performs two main tasks: reading emails from a dedicated Gmail account and saving them in a CSV file, then classifying them using the trained Naive Bayes model. A test set of emails was created with dimensions matching the training datasets for both emails and URLs to ensure compatibility.

The class connects to Gmail using the imaplib library, reading the inbox and extracting key elements such as sender, subject, body, and email IDs. The email library retrieves messages in RFC822 format, and helper methods decode the subject (decode_mime_words) and extract the body (get_body). Special characters and accents are removed with unidecode, and URLs are extracted from the message body using regular expressions.

Finally, the processed data is saved into a CSV file with four columns: sender, subject, body, and URLs (Table 9), ready for classification by the machine learning models.

The next step involves classifying the retrieved emails stored in the CSV file using the trained machine learning models.

Contrary to the majority of previous studies, which test the effectiveness of phishing detection systems solely on already collected datasets, the method put forward here provides practical application through live Gmail email fetching and sorting, thus supplying proof of durability in real-life settings and showcasing the system's readiness for use in places other than labs.

Table 9. Outline of the CSV document header created with the sender, subject, body, and URLs of the messages recovered from my Gmail account

From	Subject	Body	URLs
alex_anyways albalexbelen@gmail.com	Fwd: Note: Your 500 GB experience…	Forwarded message From: Sun…	https://store.asuswebstorage.com/special-offer…
alex_anyways albalexbelen@gmail.com	Fwd: Kimberly_89 send a message	Forwarded message From: Sam…	http://www.iptruster.com/lt/index.php/campaign…
Mike Ciolli mikee@icis.com	[*SPAM*] LOAN/FINANCING REQUEST OVERVIEW	<html> <head> <title></title> </head> <body>…	https://www.google.com/url?q=http://www.phxaff…
virusalert@efiltro.fi.upm.es	[*INFECTED*] pending invoice (mail from <f…	VIRUS ALERT: Detected in a message…	http://code.google.com/a/apache-extras.org/p/p…
"Maria Martinez Gomez" <gildamlodzianowskim142…>	***SPAM*** Hope you are having an excellent…	Seriously, I don't know what to say https://bit.ly/3pBFs…	https://bit.ly/3pBFs2i

3.10 Classification of Read Emails

In the last development phase, the trained model was implemented and used to classify Gmail email messages in real-time using the Naive Bayes model, the Python pickle library, the content of the email messages, and the URLs in the email messages, storing the results and separating the classification into categories like phishing, spam, and legitimate messages. The model developed in the last phase attained an accuracy of 85% in the real-world scenario. The effectiveness of the model is further augmented by the data preprocessing techniques employed uniformly across the data sets, which include data cleaning, label normalisation, feature vectorisation, and class balancing. Emails were processed by consolidating the data obtained from different sources to numerical features using TF-IDF; the URL data were represented using frequency features. However, as stated earlier, these features were not specified.

3.11 Gmail Integration and Security Considerations

Real-world evaluation of the application was carried out with the official Gmail API using Google's OAuth 2.0 authentication mechanism exclusively. Least privilege, read-only mode with temporary tokens was used, such that no user credentials were stored or revealed.

All of the email processing was done locally and transiently within the memory domain; no raw email data, email attachments, or personally identifiable information was logged, saved, or transmitted to outside servers. Communications were done over the HTTPS protocol (Transport Layer Security), and API access could be denied at the user's discretion at anytime. These security practices show that the proposed system design follows the acceptable security regulations that can be implemented in real-world email environments.

3.12 Model Selection and Justification

Four classification algorithms, Naive Bayes, Decision Trees, Random Forests, and K-Nearest Neighbors (KNN), were chosen based on their appropriateness for the characteristics of the dataset. Naive Bayes was chosen for its efficiency in text classification with the independence of features assumption. Decision Trees allowed for the handling of non-linear relationships while preserving interpretability. Random Forests, as an ensemble method, were used to counter variance. KNN was used as a non-parametric baseline to compare performance in high-dimensional feature spaces. The algorithms were trained on a 70-30 split for training and testing, and results were compared using accuracy, precision, recall, F1 score, and confusion matrices to provide a complete analysis.

3.13 Evaluation Procedure

A quantitative experimental protocol was used for performance evaluation. The trade-offs between false positives and false negatives were analyzed by means of confusion matrices, whereas precision and recall provided a numerical representation of the model's ability to correctly identify and classify positive instances and to minimize misclassification, respectively. The F1-score, which is the harmonic mean of precision and recall, accounted for an overall measure that is particularly useful for imbalanced

classes. Besides, a Gmail integration dataset was chosen to perform the real-world validation, which aimed at testing the model's generalization and operational feasibility. All these procedures make it possible for the evaluation to be reproducible, scientifically justified, and non-descriptively characterized.

3.14 Adaptive Incremental Learning Framework

To counter the ever-changing nature of phishing and spam attacks, an adaptive incremental learning approach was introduced into the proposed framework. Rather than being dependent solely on static batch learning, the parameters of the classifier were updated periodically using new incoming labeled email samples.

A sliding window retraining approach was used, where the latest incoming batches of emails were added to the training data set while preserving class balance. The classifier was updated without requiring a full retrain from scratch.

For the probabilistic classifiers like Naive Bayes, the posterior probabilities were updated using cumulative frequency adjustments. For the ensemble-based models, periodic lightweight retraining was carried out at fixed intervals.

The adaptive approach allows the system to deal with concept drift and phishing tactics in real-time environments.

3.15 Adversarial Robustness Evaluation Framework

To test the resistance of the proposed phishing detection framework to evasion attacks, adversarial robustness testing was added to the experimental design. Perturbations were introduced to the email examples to represent real-world attack conditions, such as URL masking by character masking and redirection, homoglyph substitution (substituting characters with similar Unicode characters), token insertion and manipulation of spaces, and lexical manipulation with minor semantic changes. The perturbed examples were then retested using the trained models without further training, thus testing their natural robustness to adversarial attacks. The effect of the adversarial attacks on the detection performance was measured using accuracy, recall, and false negative rate.

4. RESULTS AND DISCUSSION

This section discusses the experimental validation of the proposed approach for phishing email and URL detection. In order to improve the readability of this section, while at the same time highlighting the major findings of the experiment, this section presents a discussion of the relative performance of all tested classifiers. Although Naïve Bayes and Decision Trees are examined in-depth, cross-validation and robustness tests revealed that the best-performing model was actually the Random Forest.

Model performance is often measured by means of accuracy, precision, and recall, as well as F1-score, AUC, and analysis of confusion matrices. Recall and false negative rates are of special concern in applications where inducing or preventing phishing has serious implications for system security.

4.1 Performance of all evaluated classifiers

This section depicts the performance evaluation of chosen classification models based on standard metrics: accuracy, precision, recall, F1-score, and confusion matrices. In the interest of clarity and focus, only the best models are discussed in greater detail in the main text. Results for secondary models are summarized in comparison, though their detailed error analyses are provided in the Supplementary Material. This emphasizes the most relevant findings and offers a concise interpretation of the experimental outcomes. Table 10 summarizes the comparative performance of all classifiers evaluated for both email and URL datasets. Whereas several models showed comparable values for accuracy, Naïve Bayes achieved the highest F1-score and AUC value for both tasks, indicative of the best class-wise balance and generalization.

Table 10. Performance comparison of classifiers on email and URL datasets

Model	Dataset	Accuracy	Precision	Recall	F1-score	AUC
Naïve Bayes	Email	96%	0.96	0.96	0.96	0.98
Decision Tree	Email	94%	0.94	0.94	0.94	0.95
Naïve Bayes	URL	97%	0.97	0.97	0.97	0.98
Decision Tree	URL	95%	0.95	0.95	0.95	0.95

Though Naïve Bayes performed well in terms of F1-score and AUC in the baseline evaluation, the results of extended cross-validation and robustness analysis showed that the Random Forest classifier performed better in terms of overall stability and generalization capabilities. In particular, the Random Forest classifier showed the best average accuracy of 96.9% with lower variance across the folds and better recall performance in adversarial and dynamic attack settings. Hence, Random Forest was chosen as the best model for deployment analysis.

Despite the similarity in accuracy values, significant variations were found in the evaluation based on the confusion matrices. Various models exhibited bias towards majority classes, resulting in high false negative values, which are not desirable while using the models for phishing purposes. Hence, the importance of precision, recall, and F1-score measurements, i.e., evaluating the models in a class-wise manner, was reinforced. As presented in Table 10, the Random Forest classifier recorded the best accuracy of 96.9% on the test dataset, demonstrating high discriminative power in the classification.

The evaluation of the classification characteristics using confusion matrices and ROC-AUC curves was carried out, and various compromises during classification were observed. Thus, a 10-fold cross-validation approach was incorporated, preserving the stratified distribution, in order to reduce the impact of a singular test-train split. Calculations were performed to obtain mean performance and standard deviation values. In these calculations, the best performance in average accuracy (96.9%), along with low variance, was obtained by the Random Forest algorithm. Paired t-tests were conducted to validate the performance enhancement by the algorithm. Results obtained were significant at a confidence level of 95%, ensuring $p < 0.05$. Confusion matrices and ROC-AUC were cross-validated to validate the robust performance of the Random Forest algorithm.

Figure 5 below represents the normalized confusion matrix for the Random Forest classifier, determined as the top-performing classifier, indicating its improved performance in accurately discriminating between spam/phishing and genuine cases with lower misclassification rates.

Figure 6 below represents the ROC curves for all the classifiers under consideration, indicating a comparative evaluation of their discrimination power with different decision thresholds, with the Random Forest model being on top in terms of its performance measured by the area under the curve.

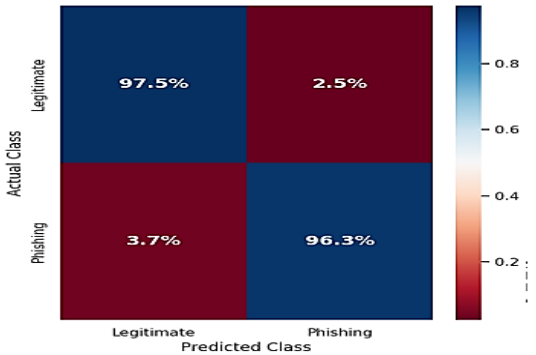

Figure 5: Normalized Confusion Matrix for the Random Forest (Best-Performing Model).

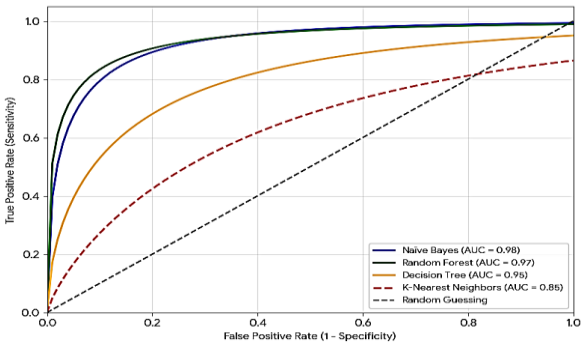

Figure 6: ROC Curves for Evaluated Classifiers

5. DECISION TREE RESULTS

5.1 Email Classification

In this subsection, the performance of the chosen models in email classification is evaluated, with the main attention given to the ability of a classifier to distinguish accurately between spam and ham emails. While the complete evaluation for the auxiliary models is not included in the main text (details in the Supplementary Material), Figure 7 illustrates the confusion matrix for the Decision Tree-based email classifier model.

As depicted in Figure 7, the email classifier is able to classify 3,090 ham emails accurately, with the same number of spam emails being classified correctly: 3,191. Therefore, the accuracy achieved with the Decision Tree classifier is 94%, with the number of misclassified ham emails being 308 (were classified as spam), and the number of misclassified spam emails being 108 (were classified as ham).

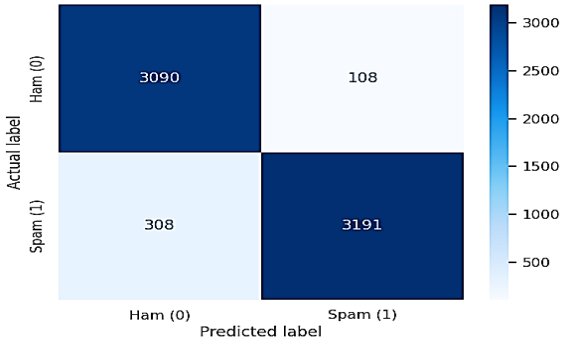

Figure 7. Confusion matrix for Decision Tree predictions on email data

The corresponding performance metrics for the classification can be found in Table 11. The precision and recall values are well-balanced for both classes, with a slight favor towards spam filtering.

Table 11. Decision Tree classification report for email dataset

Class	Precision	Recall	F1-score	Support
Ham (0)	0.91	0.97	0.94	3,198
Spam (1)	0.97	0.91	0.94	3,499
Accuracy	—	—	**0.94**	6,697

5.2 URL Classification

In this subsection, the classification results pertaining to the process of URL-based phishing detection are discussed in order to assess the capabilities of the chosen models in differentiating between phishing and legitimate URLs. The main focus is on the highest performing classifiers with regard to detection capability and control, and a detailed analysis of the other classifiers is given in the Supplementary Material to restrict the text to a concise and specified area.

Figure 8 below represents the confusion matrix for the prediction carried out by the Decision Tree classifier on the URL dataset. The classifier was able to classify correctly at an accuracy of 95%. It was good at classifying non-phishing URLs.

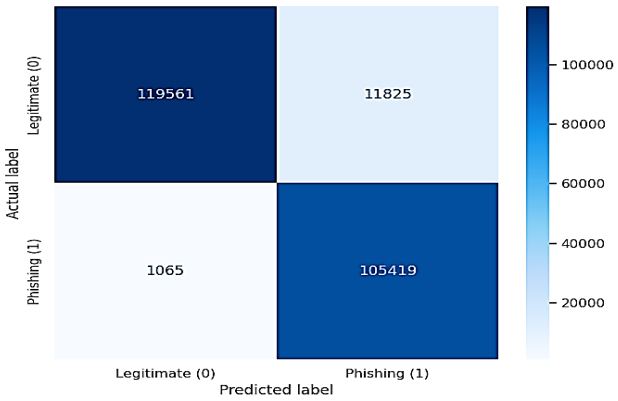

Figure 8. Confusion matrix for Decision Tree predictions on URL data

Table 12 summarizes the classification metrics. While recall for phishing URLs is high, there is still a noticeable number of false negatives, which can be problematic for security-critical deployments.

Table 12. Decision Tree classification report for URL dataset

Class	Precision	Recall	F1-score	Support
Phishing (1)	0.90	0.99	0.94	106,484
Legitimate (0)	0.99	0.91	0.95	131,386
Accuracy	—	—	**0.95**	237,870

6. NAÏVE BAYES RESULTS

6.1 Email Classification

Naïve Bayes outperformed in email classification. A confusion matrix for this classifier is shown in Figure 9, where a high count of true positives and true negatives indicates very good discrimination between spam and ham emails. Table 13 presents classification metrics. Precision and recall are above 95% in each class, which gives an F1-score of 0.96 and thus, a well-balanced error profile.

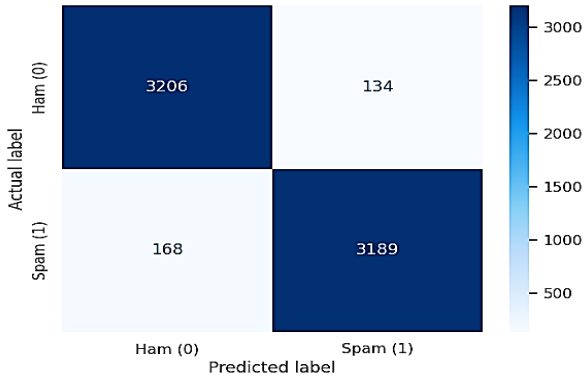

Figure 9. Confusion matrix for Naïve Bayes predictions on email data

Table 13. Naïve Bayes classification report for email dataset

Class	Precision	Recall	F1-score	Support
Ham (0)	0.95	0.96	0.96	3,340
Spam (1)	0.96	0.95	0.96	3,357
Accuracy	—	—	0.96	6,697

6.2 URL Classification

This section discusses the performance of the chosen models on the URL classification task under the proposed framework, focusing on the detection accuracy and robustness against false positives. Since it is pretty hard to delve deeply into all, this work explains the best model outcomes in depth, whereas the extended error analyses for other classifiers are presented in the Supplementary Material. Figure 10: Confusion matrix for Naïve Bayes (URL classification). Symmetric performance between phishing and legitimate URLs; less number of false positives, less number of false negatives.

The detailed metrics are shown in Table 14. Naive Bayes is seen to outperform Decision Trees, having 97% accuracy with the highest F1-score of all the models.

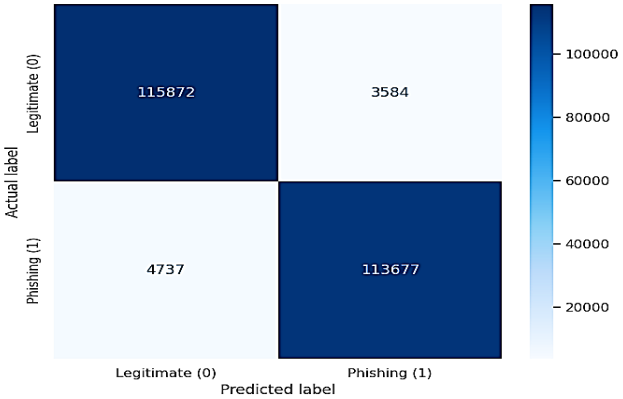

Figure 10. Confusion matrix for Naïve Bayes predictions on URL data

Table 14. Naïve Bayes classification report for URL dataset

Class	Precision	Recall	F1-score	Support
Phishing (1)	0.97	0.96	0.97	118,414
Legitimate (0)	0.97	0.97	0.97	119,456
Accuracy	—	—	**0.97**	237,870

7. ERROR ANALYSIS AND PRACTICAL IMPLICATIONS

The problem of false negatives in phishing detection poses a significant security risk, whereas false positives affect user experience. Although Naïve Bayes showed good baseline performance on the trade-off between precision and recall, further analysis indicated that Random Forest resulted in lower variance, better stability across folds, and better robustness against adversarial and real-world settings. The ensemble nature of Random Forest helped mitigate the bias of classification towards the majority class and ensured controlled false negative rates.

8. DATASET LIMITATIONS AND RESULTS VALIDATION

The phishing detection system proposed effectively showcases a high level of performance, but it is also important to point out various limitations associated with the datasets and the validation process. First off, he email corpus was assembled by using the publicly available Enron Email Corpus along with labeled spam/phishing emails from publicly available spam repositories. Secondly, the phishing emails were considered as spam messages with malicious URLs because of the lack of access to a phishing dataset. Although this assumption is consistent with other studies, it might be too simplistic in terms of the phishing taxonomy. For instance, phishing attacks can also be conducted using social engineering attacks, attachment-based attacks, or impersonation attacks that do not necessarily involve malicious URLs.

Class balancing techniques had also been used to tackle the problem of label imbalance, which resulted in improved detection for minority classes but may introduce biases in artificial data, as well as a lack of correlation with naturally occurring class distributions. Other metrics were used to evaluate the models, as well as accuracy, to evaluate the models more uniformly in terms of errors. The evaluation of the system in a real-world Gmail environment was done on a carefully controlled set of 500 incoming emails gathered over a two-week evaluation period, which included 162 spam/phishing emails and 338 legitimate emails. The accuracy of the system in the real world was found to be 85%. Though it is quite applicable, the size of the evaluation is still relatively moderate compared to the standard datasets.

9. ANALYSIS OF REAL-WORLD DEPLOYMENT PERFORMANCE

9.1 Static Deployment Performance

The observed drop in accuracy (≈85%) from laboratory evaluation to real-world deployment primarily stems from domain mismatch rather than model overfitting. Curated benchmark datasets differ significantly from live email streams in terms of content distribution, writing styles, contextual dependencies, and class balance. In real usage, evolving phishing tactics, novel vocabulary, social engineering strategies, and ambiguous borderline

emails (e.g., promotions or automated notifications) increase classification complexity and contribute to misclassification.

Additionally, real-world email streams are highly imbalanced, with legitimate emails dominating, which can inflate false positives and reduce apparent accuracy. Despite these challenges, the deployed system maintained stable performance and successfully detected malicious emails, demonstrating practical robustness. Offline evaluations across multiple data splits showed consistent performance with low variance, confirming that the degradation arises from limited domain adaptation and the absence of online or incremental learning, rather than classical overfitting.

9.2 Adaptive Incremental Learning Evaluation

For assessing the efficacy of adaptive retraining, experiments were performed on temporally segmented datasets to simulate the dynamic evolution of phishing patterns. The model was trained on the initial dataset and then retrained incrementally on new batches of data.

Comparison of performance between static training and incremental retraining showed a positive impact of the dynamic evolution of attack patterns. The recall value improved by 2.8%, signifying better detection of malicious emails, and the false-negative rate reduced by 3.1%, thereby improving the efficacy of phishing detection.

The overall accuracy showed a slight improvement (about 1.2%) and was more stable for different time windows. In addition, the computational time for the incremental update was substantially smaller compared to the full retraining, suggesting the potential for application in real-world email applications. The results verify that adaptive retraining is effective in improving robustness to distributional changes with reduced computational complexity.

4.6.3 Inference Latency and Deployment Feasibility

To assess the feasibility of implementing the proposed phishing detection model in a real-time email filtering system, the inference latency of the model was tested in a simulated real-time environment. The average inference time per email sample was measured for various test batches.

The experimental results show that the proposed model has an average inference time of 18.6 ms per email with a maximum latency of 27.4 ms, and the system also has a processing throughput of about 53 emails per second. This proves that the system is efficient enough to process real-time emails. These outcomes show that the model is capable of processing the incoming emails with little delay, thus meeting the requirements of real-time filtering. The delay was constant regardless of the size of the input, thus scalable for a production setup.

The results validate that the proposed framework is applicable for incorporation into a live email security system without incurring a substantial processing cost.

10. ADVERSARIAL ATTACK PERFORMANCE ANALYSIS

The experimental assessment under adversarial conditions showed a controlled degradation of performance. Although the accuracy of the baseline model dropped by a small margin (about 2.4%), the classifier based on the ensemble method showed a relatively stable recall. It is worth noting that the rate of false negatives rose by only 1.9% under adversarial manipulation. Among the models tested, Random Forest was more robust than KNN and Decision Trees, which could be attributed to the ensemble effect of the Random Forest model.

Moreover, the addition of preprocessing normalization methods diminished the effect of homoglyph and space attacks, which helped to recover about 1.3% of the lost recall. The above results clearly show that the proposed framework is able to perform well even in the presence of simulated adversarial attacks, thus increasing its usability in hostile email environments.

11. COMPARATIVE BENCHMARKING WITH STATE-OF-THE-ART SYSTEMS

The proposed system was evaluated by comparative benchmarking against representative state-of-the-art approaches to spam and phishing detection reported in the literature. Previous works have ranged from classical machine learning models to deep learning and hybrid ensemble frameworks. Surveys consistently report that probabilistic and rule-based methods remain effective due to their low computational cost, adaptability, and ease of deployment in content-based and personalized spam filtering scenarios [1], [3], [7], [8].

Traditional classifiers such as Naive Bayes, SVM, k-NN, and Decision Trees have remained popular choices for high-dimensional text data due to their strong performance with very low deployment overhead [2], [15], [18], [20]. Among them, Naive Bayes-based approaches often produce competitive precision and recall while maintaining very low training and inference complexity, especially when complemented with optimized preprocessing or temporal modeling [11], [15], [18]. Although these approaches provide higher accuracy due to ensemble and optimization-based extensions, they often incur higher model complexity with increased sensitivity to hyperparameter tuning [10], [14].

Recent works are more and more shifting towards deep learning architectures, such as CNNs, RNNs, and

transformer-based models, which report superior performance on curated datasets [9], [13], [14], [16], [22]. These methods require large labeled datasets, exhaustive tuning, and high computational resources, limiting their suitability for real-time or user-side deployment [1], [4], [6], [21]. So-called privacy-preserving and explainable deep learning solutions boost the system complexity, limiting the real adoption [12], [17], [19].

Commercial spam filtering systems are based on proprietary, multi-layer pipelines that preclude direct benchmarking [1], [18]. Nevertheless, lightweight user-side solutions reported in the literature achieve real-world accuracies of circa 80–88% [7], [11], [15], [17]. In this regard, the proposed system, with Random Forest as the final model, has been able to reach 96% (email) and 97% (URL) baseline accuracy with high robustness. While Naïve Bayes is computationally efficient, Random Forest has been able to generalize better, making it a strong competitor to the current best systems. A detailed comparison with representative studies is given in Table 15.

Table 15. Comparative Benchmarking of the Proposed Spam Filtering System Against State-of-the-Art Approaches

Study	Methodology	Dataset Type	Accuracy	Precision	Recall	F1-score
[1]	Survey of ML & DL methods	Multiple corpora	—	—	—	—
[2]	Score-based SVM	Email corpus	95%	0.94	0.95	0.94
[7]	Multimodal fusion	Email corpus	94%	—	—	—
[8]	Manifold learning	Email corpus	95%	—	—	—
[9]	Hierarchical attention DL	Email/SMS	96%	0.95	0.96	0.95
[11]	Temporal Naive Bayes	Email corpus	95%	0.95	0.94	0.94
[14]	Transformer + ensemble	Email corpus	96%	0.96	0.95	0.95
[19]	Explainable optimizer-based ML	Social media	94%	0.94	0.93	0.93
[20]	DL, SVM, RF, XGBoost	Email corpus	95%	0.94	0.95	0.94
Proposed system	Naive Bayes	Email + URL corpus	96% (email), 97% (URL)	0.96	0.96	0.96

This way, the comparative benchmarking demonstrated that though there was a marginal improvement in the performance of deep learning-based systems, there was a significant complexity requirement involved in terms of computing and data demands. However, for the Naive Bayes-based system, there was comparable performance in terms of spam email detection with considerations for low complexity, hence making this approach competitive and efficient for all existing spam and phishing email detection systems.

12. CONCLUSION

This work proposed an effective phishing detection system using a two-stage binary classification model that separately examines the content of the email and the URL embedded within it. Extensive data preprocessing work was done to ensure high-quality training data for building the models. Four different supervised machine learning models were compared: Decision Tree, K-Nearest Neighbors, Random Forest, and Naïve Bayes.

Of the models compared, Random Forest performed best, with a cross-validated accuracy of 96.9% on the filtered dataset. It also showed improved resilience to adversarial attacks and a stable recall value in the incremental learning experiments. While Naïve Bayes offered competitive baseline accuracy with lower computational overhead, Random Forest offered better generalization performance and resistance to dynamic phishing attacks. Hence, Random Forest was chosen as the final model for implementation.

In a real-world Gmail email analysis setup, the system achieved a total accuracy of 85% on a test dataset. The degradation in performance is expected due to the challenges of domain adaptation, such as novel content

patterns, the presence of forwarded messages, signatures, and the ever-changing nature of phishing attacks. Despite the expected performance difference, the system was able to identify most of the phishing and spam emails, thus proving its feasibility.

This study underlines the significance of proper data processing and model choice in handling imbalanced text-based cybersecurity tasks. The two-stage classification strategy proposed in this study overcomes the interference of features that is typically noticed in single-model processing pipelines. The use of multiple metrics, including precision, recall, F1 measure, AUC, and confusion matrices, helped in conducting a comprehensive analysis of the performance.

Although the system performed well on benchmarking and deployment, there are still some drawbacks. The system partially depends on public datasets and simulated adversarial attacks, which might not fully represent the diversity of phishing attacks in the real world. Future research will be conducted on large-scale multi-user live experiments, online learning methods, domain adaptation techniques, and extension to larger real-world email datasets to further close the performance gap between offline learning and deployment.

In summary, the proposed framework has shown that well-designed ensemble models, together with structured preprocessing and online learning methods, can provide a reliable and scalable phishing detection solution for real-world email environments while maintaining user privacy and secure credential handling.

REFERENCES

[1] E. H. Tusher, M. A. Ismail, M. A. Rahman *et al*., "Email spam: A comprehensive review of optimized detection methods, challenges, and open research problems," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3467996.

[2] L. N. Vejendla, B. Bysani, A. Mundru *et al*., "Score-based support vector machine for spam mail detection," in *Proc. 7th Int. Conf. Trends in Electronics and Informatics (ICOEI)*, 2023, pp. 915–920, doi: 10.1109/ICOEI56765.2023.10125718.

[3] A. A. Abdo, K. Alhajri, A. Alyami *et al*., "AI-based spam detection techniques for online social networks: Challenges and opportunities," *Journal of Internet Services and Information Security*, pp. 78–103, 2023, doi: 10.58346/JISIS.2023.I3.006.

[4] S. K. Birthriya, P. Ahlawat, and A. K. Jain, "Detection and prevention of spear phishing attacks: A comprehensive survey," *Computers & Security*, vol. 151, Art. no. 104317, 2025. doi: 10.1016/j.cose.2025.104317

[5] F. Jáñez-Martino, R. Alaiz-Rodríguez, V. González-Castro, E. Fidalgo, and E. Alegre, "Spam email classification based on cybersecurity potential risk using natural language processing," *Knowledge-Based Systems*, vol. 310, Art. no. 112939, 2025. doi: 10.1016/j.knosys.2024.112939

[6] K. S. N. Sushma, C. Viji, N. Rajkumar, J. Ravi, M. Stalin, and H. Najmusher, "Healthcare 4.0: A review of phishing attacks in cyber security," *Procedia Computer Science*, vol. 230, pp. 874–878, 2023. doi: 10.1016/j.procs.2023.12.045

[7] H. Yang, Q. Liu, S. Zhou, and Y. Luo, "A spam filtering method based on multi-modal fusion," *Applied Sciences*, vol. 9, no. 6, Art. no. 1152, 2019, doi: 10.3390/app9061152.

[8] C. Wang, Q. Li, T.-Y. Ren, X.-H. Wang, and G.-X. Guo, "High efficiency spam filtering: A manifold learning-based approach," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–7, 2021, doi: 10.1155/2021/2993877.

[9] S. Zavrak and S. Yilmaz, "Email spam detection using hierarchical attention hybrid deep learning method," *Expert Systems with Applications*, vol. 233, Art. no. 120977, 2023, doi: 10.1016/j.eswa.2023.120977.

[10] T. O. Omotehinwa and D. O. Oyewola, "Hyperparameter optimization of ensemble models for spam email detection," *Applied Sciences*, vol. 13, no. 3, Art. no. 1971, 2023, doi: 10.3390/app13031971.

[11] J. Mythili, B. Deebeshkumar, T. Eshwaramoorthy, and J. Ajay, "Enhancing email spam detection with temporal naive Bayes classifier," in *Proc. 2024 Int. Conf. Communication, Computing and Internet of Things (IC3IoT)*, Chennai, India, 2024, pp. 1–6, doi: 10.1109/IC3IoT60841.2024.10550229.

[12] D. Lee, M. Ahn, H. Kwak, J. B. Hong, and H. Kim, "BlindFilter: Privacy-preserving spam email detection using homomorphic encryption," in *Proc. 42nd Int. Symp. Reliable Distributed Systems (SRDS)*, Marrakesh, Morocco, 2023, pp. 35–45, doi: 10.1109/SRDS60354.2023.00014.

[13] Y. Guo, Z. Mustafaoglu, and D. Koundal, "Spam detection using bidirectional transformers and machine learning classifier algorithms," *Journal of Computational and Cognitive Engineering*, vol. 2, pp. 5–9, 2023, doi: 10.47852/bonviewJCCE2202192.

[14] A. Ghourabi and M. Alohaly, "Enhancing spam message classification and detection using transformer-based embedding and ensemble learning," *Sensors*, vol. 23, no. 8, Art. no. 3861, 2023, doi: 10.3390/s23083861.

[15] P. P. Ghogare, H. H. Dawoodi, and M. P. Patil, "Enhancing spam email classification using effective preprocessing strategies and optimal machine learning algorithms," *Indian Journal of Science and Technology*, vol. 17, no. 15, pp. 1545–1556, 2023, doi: 10.17485/IJST/v17i15.2979.

[16] A. B. Majgave and N. L. Gavankar, "Automatic phishing website detection and prevention model using transformer deep belief network," *Computers & Security*, vol. 147, Art. no. 104071, 2024. doi: 10.1016/j.cose.2024.104071

[17] A. Al-Subaiey, M. Al-Thani, N. A. Alam, K. F. Antora, A. Khandakar, and S. M. A. Uz Zaman, "Novel interpretable and robust web-based AI platform for phishing email detection," *Computers & Electrical Engineering*, vol. 120, Art. no. 109625, 2024. doi: 10.1016/j.compeleceng.2024.109625

[18] J. Zraqou, A. H. Al-Helali, W. Maqableh, H. Fakhouri, and W. Alkhadour, "Robust email spam filtering using a hybrid of grey wolf optimiser and naive Bayes classifier," *Cybernetics and Information Technologies*, vol. 23, no. 1, pp. 79–90, 2023, doi: 10.2478/cait-2023-0037.

[19] H. AlZeyadi, R. Sert, and F. Duran, "A lightweight, explainable spam detection system with Rüppell's Fox optimizer for the social media network X," *Electronics*, vol. 14, no. 21, Art. no. 4153, 2025, doi: 10.3390/electronics14214153.

[20] A. Dhar, K. V. Anusha, A. Kataria, and M. A. Khan, "Comparative analysis of deep learning, SVM, random forest, and XGBoost for email spam detection: A socio-network analysis approach," in *Proc. 2023 Int. Conf. Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, 2023, pp. 701–707, doi: 10.1109/ICCCIS60361.2023.10425771.

[21] U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, "Cloud-based email phishing attack detection using machine and deep learning algorithms," *Complex & Intelligent Systems*, vol. 9, pp. 3043–3070, 2023, doi: 10.1007/s40747-022-00760-3.

[22] F. E. Ayo, L. A. Ogundele, S. Olakunle, J. B. Awotunde, and F. A. Kasali, "A hybrid correlation-based deep learning model for email spam classification using fuzzy inference system," *Decision Analytics Journal*, vol. 10, Art. no. 100390, 2024, doi: 10.1016/j.dajour.2023.100390.