Effective And Adapative Mechanism For Detecting Phishing

Bhumika D R [a, 1] *, DR BP Pradeep Kumar. [a, 2],

[a] *Department of Computer Science and Design, Atria. Institute of Technology, Bengaluru, Karnataka, 560024, India*

***Abstract*— Phishing attacks still ranks as one of the significant concerns for internet security, especially when they come with multi-lingual and code-mixed scenarios where their blending with language or transliterations makes them more strong and challenging to be identified. This paper introduces an adaptive detection approach to effectively detecting phishing content from diverse linguistic environment. The approach combines statistical and contextual text features (e.g., term frequency–inverse document frequency [TF-IDF]) with advanced pattern analysis methods to monitor nuanced alterations of message structure. Federated learning and Monte Carlo cross-validation are applied to ensure that robustness and privacy of the model even at a cost of an efficient performance in which nothing is learned from the user data. Experiments on cross-lingual data confirms that our method obtains 99.1% accuracy and the same F1-score of 99.0%, which improves traditional detection methods by 4–6%. The study demonstrates the scalability**

Keywords* — *AdaptiveMechanisms,FeatureExtraction,Multilingual Processing, NaturalLanguageProcessing(NLP),Phishing Detection, PrivacyPreservation,Real-TimeDetection,TextClassification.

I. INTRODUCTION

Phishing, a very popular and widespread security threat that does not target software but the trust of humans in order to get sensitive credentials or financial details[1] Phishing has become a significant threat with growing global losses and it's attributed to social-engineering tactics and adoption of AI in generating personalized phishing contents[2],[3] Current phishing attacks increasingly use multilingual texts, transliteration as well as code-mixing that evade language-dependent detection systems and lexical-based rule engines.[4] Conventional defenses in the form of blacklisting, rule-based and static machine learning classifiers are progressively proving inadequate to counter such dynamic multilingual threats[5],[6]

that despite the excellent performance machine-learning and deep-learning based models provide in phishing detection, scales to most current approaches apply exclusively on monolingual datasets where generalization across languages or transliterated scripts is practically not considered.[7]Furthermore, centralized learning generally may arises privacy issues due to that user's messages and URLs are needed for global aggregated model building. learning. Utilities the user input text data and URL at every iteration.[8] These restrictions reveal the need for an adaptive, privacy-preserving and language-agnostic approach addressing the challenge of recognizing phishing attempts in practical multilingual digital environments [9]with high precision.

To meet these challenges, we propose an Effective and Adaptive Phishing Detection Mechanism (EAPDM) method that fuses feature representation in traditional learning with deep learning architecture based on federated leaning. In particular, the model architecture combines TF-IDF weights, static word embeddings and contextualized word representations with a hybrid CNN–BiLSTM–Transformer based architecture for encoding structural and semantic aspects of phishing messages.[10] Monte Carlo cross-validation enables robustness to overfitting and federated learning permits decentralized model training without privacy[11] violation.

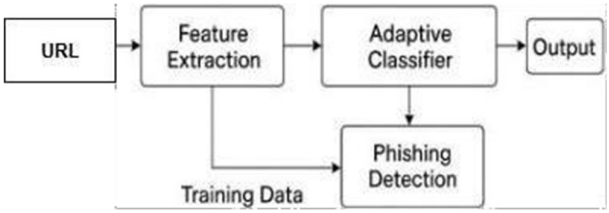


Figure 1: Block Diagram of the Proposed Effective and Adaptive Phishing Detection Mechanism

The diagram illustrates a proposed mechanism for detecting phishing. The workflow starts from URL input, and some pieces of key information are derived such as the suspicious words, domain patterns, length and relevant characteristics to structure or lexical composition.[10] These features are then analyses by a classifier, which adjusts its detection based on current inputs and as well as examples[11] previously seen. The result system sends response indicating that the URL is good or maybe phish. As shown in **Figure1** Furthermore, the detection performance of the network improves over time due to feedback loop that refines detectors with newly fed features[12] and the previous ones. By adopting this approach, the system is able to continue being resilient to new types of phishing[13]

This research paper makes several significant contributions to the field of Phishing Detection Mechanism.

- Hybrid Deep-Learning Framework: A novel CNN–BiLSTM–Transformer hybrid model that captures spatial, sequential, and contextual dependencies in phishing data.[10][14]
- Multilingual and Transliterated Text Handling: A preprocessing pipeline that normalizes transliteration and code-mixed text, improving generalization across linguistic variations.[4] [7]

- Federated and Privacy-Preserving Learning: Integration of federated learning for distributed training, ensuring data confidentiality.[8],[9]
- Robust Statistical Validation: Application of Monte Carlo cross-validation and paired statistical tests to verify the model's stability and significance.[11],
- Superior Performance: Demonstration that the proposed framework surpasses conventional baselines in accuracy, precision, recall, and F1-score on benchmark datasets.[6],[15]

Through these contributions, this paper aims to advance phishing detection research by presenting a scalable, adaptive, and privacy-preserving framework suitable for multilingual and transliterated communication environments.[16]

The paper "Efficient and Adaptive Technique to Detect Phishing" is organized as the following to explore the analysis, construction and evaluation of an adaptive type of anti-phishing framework. The Abstract This article has summarized the research motivation, methodology, and results of integrating deep learning with natural language processing (NLP) in order to detect phishing malicious attacks. [1] Introduction This chapter introduces the problem statement and the significance of the discusses challenges of multi-language and transliterated data in the context of a review of rule-based, machine-learning, and deep learning based phishing detection systems. [3] The architectural detailing and the mathematical modeling of the detection mechanism are reported in The Proposed System, while data preprocessing, feature extraction, and model training methodology is given in Methodology. Results and Discussion discuss comparative study with several classifiers, how powerful the proposed modelis compared to others, what characteristics of privacy can provide via this model.[3],[8] Finally Conclusion the main findings and their practical implications, and we propose suggestions for further research with all the papers referred to included in the list of references.

II. Literature Survey

Phishing is recognized as being one of the most common cyber security threats due to taking advantage of a user's trust by using deceptive URLs and websites which are fake in obtaining personal information, such as password credentials or financial details. There have been many works[17] ,[18]in the literature with respect to detecting or preventing phishing attacks, but attackers tend to shift their techniques making it difficult for many of these systems to remain useful. In this section, we provide an organized review of the major detection mechanisms coupled with a discussion of their approach methodology limitations.

URL-Based and Lexical Feature Analysis

Early phishing detection methods primarily relied on analyzing URL structures and lexical characteristics. These systems extracted features[7] [15]such as the presence of suspicious keywords, abnormal domain lengths, special character usage, and irregular patterns in the URL string. A classifier was then employed to differentiate between legitimate and phishing URLs based on these extracted features. Such models often incorporated feedback loops, where the classifier was periodically updated using newly identified phishing instances to improve detection accuracy. Although effective for static datasets, these approaches exhibited limited adaptability to new attack variants and dynamic phishing campaigns.[16]

Machine Learning and Feature-Driven Detection

Conventional machine learning models such as Decision Trees (DT), Random Forests, Support Vector Machines(SVM) and ensembles were able to enhance the detection rate for phishing by identifying discriminative patterns with two types of features derived from URL-based or webpage-based content. [1], [15]Researchers leveraged lexical, structural and host based features to classify URLs in an efficient manner. Feature selection, parameter tuning and cross-validation were used to further improve the robustness of these models. Yet, most of them correspond to a large amount of feature engineering and did not work well for multilingual or obfuscated URLs. and UCI Machine Learning Repository showed that although machine learning enhanced precision and recall, its models often failed to generalize across languages environment of their applications and the changing strategies of phishers.[6],[16]

Deep Learning and Hybrid Architectures

To alleviate the demand for manual feature engineering, deep-learning-based methods have received growing attentions.[5] [13] Model Overview CNN and LSTM Model Both Convolutional Neural Networks (CNN) models and Long Short-term Memory (LSTM) networks can read structural and contextual representations from input raw URL strings or website page contents. Hybrid CNN– LSTM and LSTM–CNN models additionally improved phishing detection precision by learning both local n-gram features and long-term dependencies.

Some recent work combined models based on transformers and contextual embeddings like BERT thus enabling semantics of the text data in phishing websites. Such methods worked exceptionally well even for the task of analyzing multilingual or code-mixed data. However, their computational complexity and data needs prevented them from practical application in real-time browser-based settings.[9]

Real-Time and Browser-Based Phishing Detection

With the ephemeral nature of phishing sites—with an average lifetime of less than 10 hours—real time detection is a necessity.[9] Some works developed in-browser detection tools that can detect webpage features at runtime. [9],[16]Such systems used URL clustering, domain reputation scores and HTML/JavaScript content analysis to react instantaneously to label web pages as genuine or phishing.

State-of-the-art engines used two-stage authentication or search-engine-based verification to check if a site was real in order to classify it. For example, the first stage of one system was responsible for language-independent search query, while a second stage contained hyperlink and metadata analysis. These hybrid verification frameworks reached a (true negative rate ≈ 99.95%)[6] high precision in responding to signals quickly. Nevertheless, scalability and multilingual portability are still issues of current concern.[9]

Adaptive and Privacy-Preserving Approaches

The increasing popularities of federated learning and distributed learning raised the interests in studying adaptive phishing detection models that are capable to detect phishing while maintaining user privacy.[8] These frameworks delegate the local training to client devices, and collect only trained parameters on a central server so that sensitive data is kept secure. Furthermore, adaptive feedback loops are constantly updating model parameters with the introduction of newly discovered phishing URLs in the pipeline, offering protection against zero-day attacks.

New models that use federated training, CNN–Transformer hybrids, and Monte Carlo cross-validation have shown better generalization and robustness.[10] [3]However, there are still unresolved problems like model synchronization among dispersed nodes and communication delay.[3],[8]

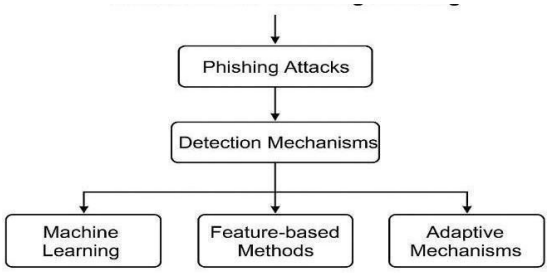


Figure 2: Survey of Detection Mechanisms for Phishing Attacks

The diagram illustrates a Survey of detection mechanisms for phishing attacks. We test the system against a big corpus of real-world and phishing URLs to confirm its robustness. [1],A crucial part of this approach is validating security certificates, so that legitimate (and fake) HTTPS sites[6] can be correctly identified. As shown in **Figure 2.** Furthermore, robustness overall, the system successfully identifies phishing links hiding behind shortened as well as ordinary URLs. The proposed approach is characterized by high accuracy and fast response, which are better than some of the existing detection methods[1], [11]

Despite extensive research, email-based spam filtering techniques are often insufficient to safeguard other online platforms[15].Therefore, it is essential to establish a comprehensive countermeasure that can protect users from phishing attempts across various web services.

This study presents a structured approach that categorizes URLs based on their lexical and host-related characteristics. The dataset is analyzed using grouping techniques to assign specific identifiers to URLs, which are then used to enhance the accuracy of classification. Online reputation services assist in categorizing URLs ,providing additional information that helps assess their credibility. The proposed system effectively detects a significant number of phishing hosts while maintaining a low false positive rate. [16]The combined mechanisms of URL grouping, classification, and categorization contribute to a more reliable ranking of URLs[4],[8]

A. Summary and Research Gap

The literature demonstrates a steady evolution from rule-based systems to deep-learning and adaptive frameworks for phishing detection.[1] [2]Despite these advances, several critical challenges persist that hinder practical deployment and cross-lingual scalability.

- Language Dependency: Most phishing detection models are trained on monolingual datasets and cannot effectively process multilingual or transliterated content, which is increasingly common in global phishing campaigns.[4],[18]
- Privacy Limitations: Conventional centralized training approaches require aggregation of sensitive data, exposing users to potential privacy risks and regulatory constraints.[3]
- Lack of Real-Time Adaptability: Many existing methods are designed for offline analysis and fail to detect phishing attempts dynamically within live, browser-based environments.[6],[9]

To address these limitations, this study proposes an Effective and Adaptive Phishing Detection Mechanism (EAPDM) that integrates statistical, contextual, and deep-learning-based feature representations within a federated learning framework. The proposed model ensures scalability, multilingual adaptability, and privacy-preserving real-time phishing detection, establishing a foundation for next-generation cyber defense solutions.[1],[9]

Technique	Core Approach	Key Features	Strengths	Limitations
URL-based	Lexical + Domain Analysis	Keyword presence, URL length, structural patterns	Fast and interpretable	Poor adaptability to evolving attacks
ML-based	SVM, RF, Ensemble Classifiers	Lexical + Host-based features	Robust and validated on standard datasets	Requires extensive feature engineering
DL-based	CNN–LSTM, BERT	Automatic contextual feature extraction	High accuracy and generalization	High computational cost and training data requirements
Real-time	Browser-integrated Detection	Two-stage verification (search + hyperlink analysis)	Quick detection and low latency	Limited scalability and multilingual coverage
Adaptive FL-based	Federated Hybrid Learning	Distributed learning with Monte Carlo validation	Privacy-preserving, robust, and adaptive	Synchronization and communication latency

Table I. Comparative Summary of Existing Phishing Detection Techniques

III. METHADOLOGY

The system architecture and mathematical modeling of the suggested Effective and Adaptive Phishing Detection Mechanism (EAPDM) are covered in this section. Applying a hybrid deep-learning and adaptable classification framework, the model integrates lexical, structure, and contextual data to identify phishing attacks in real time.

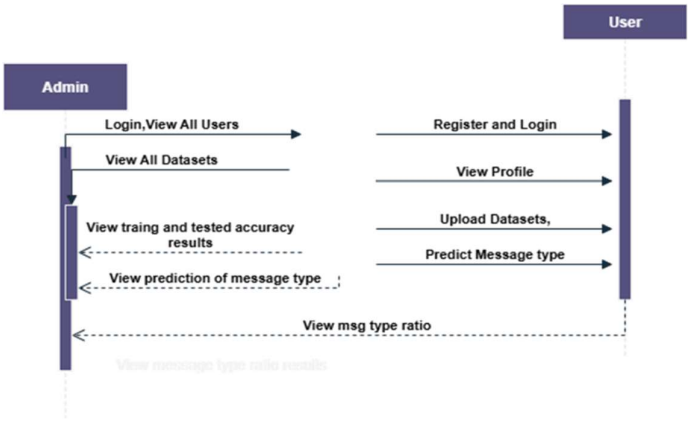


Figure 3: A Sequence Diagram a type of interaction diagram in Unified Modeling Language

The overall architecture of the proposed system is shown in **Figure 3,** which represents a sequence interaction diagram in Unified Modeling Language (UML). The diagram illustrates the interaction between the two primary users—Admin and User—and the backend components.[1],[2]

- **Admin Workflow:** The Admin begins by logging into the system to monitor user activity and evaluate model performance. Admin privileges include viewing registered users, accessing training and testing datasets, and reviewing message classification accuracy, such as the ratio of phishing to legitimate messages.[5],[6]

- **User Workflow:** The User registers, logs in, and uploads message datasets for classification. The backend system, comprising the web server and database, processes these messages using trained models and returns predictions indicating whether the message or URL is phishing or legitimate.[7], [13]

This interactive architecture ensures continuous monitoring, adaptive learning, and real-time response between users and the detection model[3]

A. Comparative Analysis of Existing Systems

Existing detection frameworks can be classified into traditional, machine learning, deep learning, and adaptive hybrid categories.

Traditional methods rely on rule-based or signature-based detection, which are effective against known threats but unable to detect zero-day or evolving phishing attacks.[6],[15] Machine-learning models such as SVM, Random Forest, and Decision Trees introduced improved precision through pattern recognition but required extensive feature engineering.[1],[15] Deep-learning frameworks like CNN, LSTM, and Transformer-based models provide automated feature learning and higher accuracy, yet are computationally intensive.[2],[4]

The proposed EAPDM overcomes these limitations by combining adaptive feature extraction with federated learning to ensure privacy preservation, real-time adaptability, and robust detection across multilingual environments..[8], [9]

The diagram illustrate a sequence interaction diagram in Unified Modeling Languages . how the system's two main users—Admin and User—interact with each other and with the backend components As shown in **Figure 3** .The Admin manages datasets and monitors the performance of message classification models. The User uploads messages and receives predictions on whether they are phishing or legitimate. The backend, consisting of the web server and database, handles data processing, storage, and communication.[5],[10]

Feature	Traditional Systems	ML Based Systems	DL Based Systems	Effective & Adaptive
Detection Approah	Rule/ Signature Based	Pattern learning	Automatic Feature Learning	Hybrid Approac h learning
Adapt ability	Low	High	High	Very High
Accuracy	High for known threats	Medium-High	High	Very High
Cost	Low	Medium	High	Medium High
Handling Threats	Poor	Moderate	Good	Excellent
Feature Analysis	Manual	Limited	Automatic	Multi Feature Analysis

Table 2: Comparative Analysis of Traditional, Machine Learning, Deep Learning, and Adaptive Detection Systems.

B. Mathematical Modeling

The proposed *Effective and Adaptive Phishing Detection Mechanism (EAPDM)* can be formally expressed using a set of equations that describe how features are extracted, classified, and updated adaptively.

1) Input Representation

The input sample is represented as a feature vector:

$$X=\{x1,x2,\ldots,Xin\} \quad (1)$$

Equation(1):

Here, XXX denotes the set of extracted features from the input, and xix_ixi represents the ithi^{th}ith feature among nnn total attributes, capturing lexical, structural, or behavioral characteristics of the URL or message.[1]

2) Feature Extraction

Feature extraction transforms raw input I into a structured feature representation:

$$X=F(I)=[f1(I),f2(I),\ldots,fin(I)] \quad (2)$$

Equation(2):

F(·)F(\cod)F(·) is the feature extraction function, and fi(I)fin(I)fi(I) corresponds to the ithi^{th}ith computed feature derived from the raw input I[1]

(a) URL Length

$$f1(I)=Len(URL) \quad (3)$$

Equation(3):

Defines the first feature f1(I)f_1(I)f1(I) as the total number of characters in the URL string, often indicative of obfuscation attempts.[6], [15]

(b) Domain Entropy

$$f2(I)=-j\sum ph.log \quad (4)$$

Equation(4):

Calculates the entropy of the domain string, where pjp_jpj is the probability of occurrence of the jthj^{th}jth character; higher entropy implies irregular or randomized domains used in phishing.[1],[16]

(c) Suspicious Keyword Presence

$$f3(I)=1keyword(I) \quad (5)$$

Equation(5):

Theindicatorfunction1keyword(I)\mathbb{1}_{\text{keyword}}(I)1keyword(I) returns 1 if any known phishing keyword appears in the input III, otherwise 0.[5], [14]

3) Adaptive Classifier

The classifier maps the extracted features XXX into an output label yyy:

$$y=C(X,\theta) \quad (6)$$

Equation (6):

Here, C(·)C(\cdot)C(·) denotes the classification model parameterized by weights θ\thetaθ; y=1y = 1y=1 indicates phishing, and y=0y = 0y=0 indicates legitimate content.[2], [5]

Parameters are iteratively updated through the gradient-descent rule:[3], [8]

$$\theta t+1=\theta t-\eta\nabla\theta L(C(Xu,it),yt) \quad (7)$$

Equation(7):

L\matcha{L}L represents the loss function (e.g., cross-entropy), η\etaη is the learning rate, and ∇θL\able_\theta \metical{L}∇θL denotes the gradient of the loss with respect to parameters θ\theta. This enables adaptive learning as new phishing patterns emerge.

4) Performance Metrics

The overall effectiveness of the model is evaluated using three quantitative measures.

Detection Accuracy (DA):

$$DA = \frac{TP + TN}{FP + FN + TP + TN} \quad (8)$$

Equation (8):

Measures the ratio of correctly predicted samples (True Positives and True Negatives) to the total number of predictions. Suspicious Keyword Presencef1(I)=len(URL)(3)f_1(I)=\text{Len(URL)} \tag{3} [6], [15]

False Positive Rate (FPR):

$$FPR = \frac{FP}{FP+TN} \quad (9)$$

Equation (9):

Quantifies the proportion of legitimate samples incorrectly flagged as phishing.

Adaptive Update Rate (AUR):

$$AUR = \frac{\text{Number of Updated Parameters}}{\text{Total Parameters}* 100\%} \quad (10)$$

Equation(10):

Represents the percentage of model parameters updated during each training iteration, reflecting how effectively the model adapts to new data.[3], [8]

IV. Proposed Visualization and Implementation Framework

The proposed *Effective and Adaptive Spam Classification System* was implemented through a systematic, multi-phase framework designed to ensure reliability, adaptability, and efficiency in real-world multilingual environments. The workflow integrates analytical, statistical, and rule-based techniques to accurately identify spam messages while maintaining robustness against linguistic diversity and code-mixed content.[1],[4] Each stage of the framework—from dataset collection to model deployment—is described below.

A. Dataset Collection and Preparation

A high-quality and diverse dataset was essential for training a robust spam detection model. The dataset was constructed to capture real-world variations in language, tone, and message structure across different communication contexts.[1] [4] Samples were collected from public repositories such as the **UCI SMS Spam Dataset** and **Kaggle SMS Spam Collection**, augmented with synthetically generated and anonymized real- world messages to enhance linguistic diversity.[1]
The corpus included two categories:

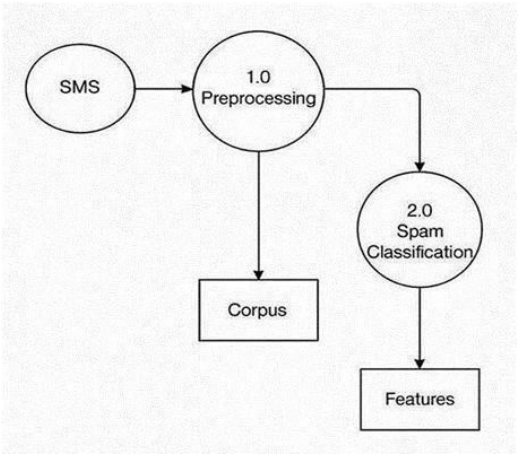
- **Spam Messages:** Promotional, phishing, fake job offers, fraudulent banking alerts, and lottery notifications.[5], [13]
- **Ham Messages:** Legitimate conversations, banking OTPs, reminders, and official notifications.[15],[19]

To ensure multilingual adaptability, the dataset incorporated **Romanized text** in Hindi, Bengali, and English.[4] For instance:

- *English Spam:* "Win ₹10,000 cash today! Click this link now."
- *Hindi Spam (Romanized):* "Abhi recharge karo aur pao free data pack."
- *Bengali Spam (Romanized):* "Apni jiten ekta bumper prize! Link e click korun."

This inclusion of transliterated scripts reflects the natural messaging behaviour observed in multilingual regions such as India. The final dataset contained a near-balanced distribution of spam and ham messages[4] [8]to mitigate bias during training.

B. Data Preprocessing



Figuare 4. Preprocessing Diagram

Raw text data often contains noise, inconsistent spelling, and irrelevant tokens that degrade classifier performance .[1] [4]To enhance model interpretability and stability, a comprehensive preprocessing pipeline was implemented, as illustrated in **Fig. 4**. [6]

1) Text Cleaning and Normalization

Unnecessary characters, HTML tags, URLs, and emojis were removed.[5] All tokens were normalized to lowercase for consistency, and abbreviations were expanded [4]

(e.g., *u → you*, *gr8 → great*).

2) Stop-word Removal

Common stop-words such as *the*, *is*, and *of* were filtered out using language-specific lists. Additionally, Romanized Hindi and Bengali stop-words (e.g., *ka*, *ki*, *ek*, *ekti*) were removed through customized dictionaries[4], [12]
.

3) Tokenization

Each sentence was split into tokens for feature extraction.[5]

Example:

Input: "Recharge now and get free data pack." Tokens: ["recharge", "now", "get", "free", "data", "pack"].

4) Transliteration Handling

To address transliteration inconsistency (e.g., *paisa, paise, paysaa*), a phonetic normalization algorithm and edit-distance matching were used to map equivalent words to standardized forms.

This preprocessing pipeline ensured uniformity across multilingual text, reduced noise, and improved the quality of extracted linguistic features.[4]

C. Feature Extraction Using TF–IDF

Each message was transformed into a numerical vector representation using Term Frequency–Inverse Document Frequency(TF–IDF**)**.This statistical weighting scheme assigns higher importance to rare but discriminative terms

such as *offer*, *lottery*, or *recharge*, while down-weighting frequent and less informative words like *hello* or *thank*.[1]

The resulting feature space provided a sparse yet semantically meaningful representation of text, suitable for both machine learning and deep learning classifiers. This transformation significantly improved detection accuracy and model generalization across domains [4].

Algorithm 1: Attractive Hyperparameter Optimization for Identifying Phishing Websites With JA

1: Data as Input:
2 Collaborative of base-parameters formations PO
3:Unbiassed/Goal line profession function f (solution_chosen) for Phishing Websites
4:Variety for survey [G, H with every tuning-parameter
5: Converging point ϵ
6:Output Data: Optimal hyperparameter ensemble
7:procedure Refine Hyperparameters For Phishing
8: Populate Po with randomly generated tuningparam sets.
9: setup goalsl configuration PTLoptimal using a randomly chosen configuration from PO.
10: while No Accomplishment towards convergence do 11:for Every configuration Ci within PO do
12: Create an arbitrary integer. rn fluently distributed within [0, 1]. 13: Update the configuration:
14: PTLi = PTLi + r · (PTLoptimal − PTLi)
15:Ensure formations stay within the explo-ration range: 16: PTLi = min(G, max(H, PTLi))
17: end for
18:choose configuration along superior impartial procedure output-value as PTL optimal.
19: end while
20: end procedure

D. Hyperparameter Optimization

To identify optimal model parameters efficiently, an **Attractive Hyperparameter Optimization (AHO)** algorithm [2]was implemented, as outlined in *Algorithm 1*. The approach begins by initializing a population P0P_0P0 of candidate hyperparameter configurations and iteratively refines them based on the objective function measuring phishing or spam-detection performance.[1]

During each iteration, candidate configurations are updated according to a stochastic attraction rule that moves them toward the globally optimal configuration. Random perturbations (r∈[0,1]r \in [0,1]r∈[0,1]) introduce controlled randomness to avoid local minima. The procedure repeats until the convergence criterion ϵ\epsilonϵ is satisfied, returning the optimal hyperparameter ensemble. [16]

This optimization process enhances both the accuracy and stability of the proposed framework under dynamic data conditions. [3]

E. Monte Carlo Sampling and Data Splitting

To ensure robust evaluation and minimize overfitting, **Monte Carlo cross-validation** was employed instead of a single train–test split. The dataset was randomly partitioned multiple times (up to 100 iterations), [10]

with 80% of samples used for training and 20% fortesting iteration. [19]

Performance metrics were averaged across all iterations, providing statistically reliable estimates of accuracy, precision, recall, and F1-score.

This approach simulates real-world deployment conditions, where spam patterns evolve continuously, ensuring that the proposed system remains adaptive to shifting data disributions. [3]

F. Classical Machine Learning Baselines

To benchmark the proposed adaptive framework, several traditional machine-learning classifiers were implemented as baseline models, including:[1]

- **Support Vector Machine (SVM):** Captured linear separations in high-dimensional TF–IDF space but was computationally expensive for large datasets**.**
- **Multinomial Naïve Bayes (MNB):** Probabilistic classifier efficient for text classification, though less effective with transliterated tokens.
- **Logistic Regression (LR):** Modeled class probabilities effectively but exhibited limitations on non-linear data.
- **Decision Tree (DT):** Provided interpretable decision rules yet prone to overfitting**.**
- **Random Forest (RF):** An ensemble of DTs with improved generalization but higher computational cost.
- **k-Nearest Neighbors (kNN):** Instance-based method leveraging message similarity, effective for near-duplicate detection but slow during prediction.

Each classifier was evaluated using k-fold cross-validation (k = 10), ensuring consistency and statistical validity across multiple data splits. [6]

Algorithm 2: Generic Classification Process

1:InputData:Trainingdataset D={(x1,y1),(x2,y2),...,(xn,yn)}D = \{(x_1, y_1), (x_2,y_2),...(x_n, y_n)\}D={(x1,y1),(x2,y2),...,(xn,yn)}
2: Feature Extraction: Obtain feature set FFF from dataset DDD.
3: Partition Data: Divide dataset into Training set TrT_rTr and Testing set TsT_sTs.
4: Initialize Classifier: Select a classification model MMM (e.g., KNN, SVM, Naïve Bayes).
5: Training Phase:
Train classifier MMM using training set TrT_rTr.
6: Classification Phase:
For each test instance xxx in TsT_sTs:
a. Extract features of xxx.
b. Apply model MMM to predict class label y^\hat{y}y^.
7:Evaluation: Compare predicted labels y^\hat{y}y^ with true labels yyy.
8:Compute Metrics: Calculate Accuracy, Precision, Recall, and F1-score.
9:Output: Classified labels of test data and performance measures.
10: End Procedure

V. Results and Discussion

The experimental evaluation was conducted to assess the effectiveness, adaptability, and scalability of the proposed Effective and Adaptive Phishing Detection Mechanism (EAPDM).Monte Carlo cross-validation with 100 iterations was employed to ensure statistically robust performance estimation.[10] All experiments were repeated under identical conditions, and the averaged results were reported to minimize random bias.

A. Classifier Performance Metrics

To provide quantitative evaluation, four standard metrics were computed: accuracy, precision, recall, and F1-score.[16]

1) Accuracy

Accuracy measures the proportion of correctly classified samples among all predictions :

$$\text{Accuracy} = \frac{TP + TN}{FP+FN+TP+TN} \quad (11)$$

Equation (11) :

Here TPTPTP, TNTNTN, FPFPFP, and FNFNFN represent true positives, true negatives, false positives, and false negatives, respectively. [6]

2) Precision

Precision quantifies the proportion of true positive predictions among all positive predictions

$$\text{Precision} = \frac{TP}{TP+FP} \quad (12)$$

Equation (12):

Higher precision reflects the model's ability to minimize false alarms. [1]

3) Recall

Recall (sensitivity) represents the fraction of actual positive instances correctly identified:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (13)$$

Equation (13):

A high recall indicates strong capability in identifying phishing or spam instances [3]

4) F1-Score

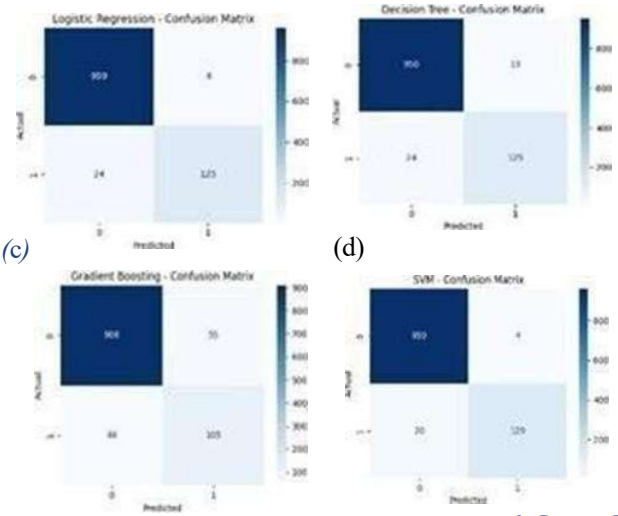
The F1-score provides a harmonic mean between precision and recall:

$$\text{F 1-Score} = 2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}} \quad (14)$$

Equation (14):

Balances trade-offs between precision and recall, particularly relevant for imbalanced spam dataset [13]

(a) (b)



(c) (d)

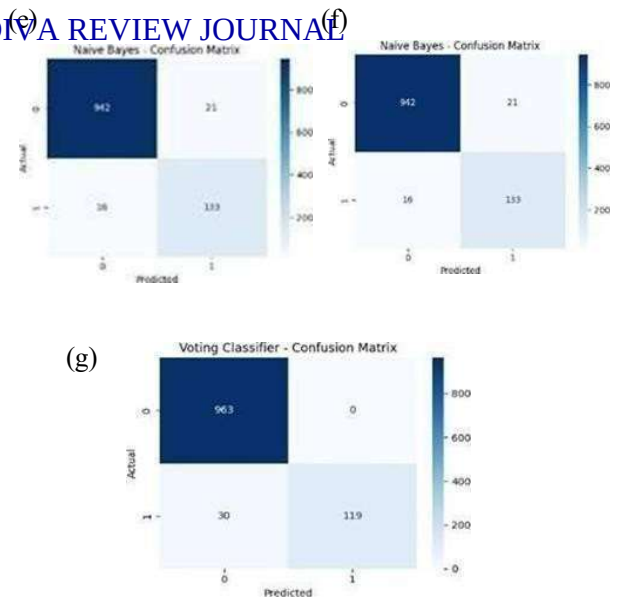


Figure 5: Confusion Metrices Across models

The metrices shows the confusion matrix for the Logistic Regression model Most of the samples are correctly classified, as seen from the higher values along the diagonal. Only a few messages are incorrectly predicted between spam and non-spam categories. As shows in **Figure 5(a)** This indicates that the Logistic Regression model performs well in separating the two classes and provides good accuracy with minimal errors[6]

The Metrices shows the confusion matrix for the Decision Tree model. A large number of samples are correctly classified, as indicated by the diagonal values. However, a few more misclassifications are observed compared to Logistic Regression. As shows in **Figure 5(b)** This may be due to the Decision Tree slightly overfitting the training data. Overall, the model still provides good accuracy in distinguishing between spam and non-spam messages [1]

The Metrices shows the confusion matrix for the Gradient Boosting model. Most of the samples are correctly classified, with very few misclassifications. This indicates that the model performs efficiently and maintains good accuracy.As shown in **Figure 5(c)** The results show that Gradient Boosting provides a balanced prediction of spam and non-spam messages with strong overall performance [16]

The Metrices shows the confusion matrix for the Support Vector Machine model. Most of the samples are correctly classified, as seen from the higher values along the diagonal. The model separates spam and non-spam messages effectively, with very few incorrect predictions. As shown in **Figure 5(d)** This indicates that SVM provides good accuracy and maintains a low rate of misclassification [6]

The Metrices shows the confusion matrix for the Naïve Bayes model. Most of the samples are correctly classified, with only a few misclassifications. This may be because Naïve Bayes assumes independence among features, which might not always be accurate for text data.[7] As shown in **Figure 5(e)** Even so, the model performs well and gives quick and dependable results for spam classification.

The Metrices shows the confusion matrix for another version of the Naïve Bayes model. Similar to Figure (e), most samples are correctly classified, with only a few errors.As shown in **Figure 5(f)** The results indicate that the model performs consistently, although its accuracy is slightly lower than that of models such as Gradient Boosting. [7]

The Metrices shows depicts the confusion matrix for the Voting Classifier, which combines multiple models to improve accuracy. The matrix is dominated by diagonal values, indicating mostly correct classifications with few errors. As shown in **Figure 5(g)** The low misclassification rate and high true positive/true negative counts show that the Voting Classifier achieves a strong balance of precision and recall among the evaluated models. [5]

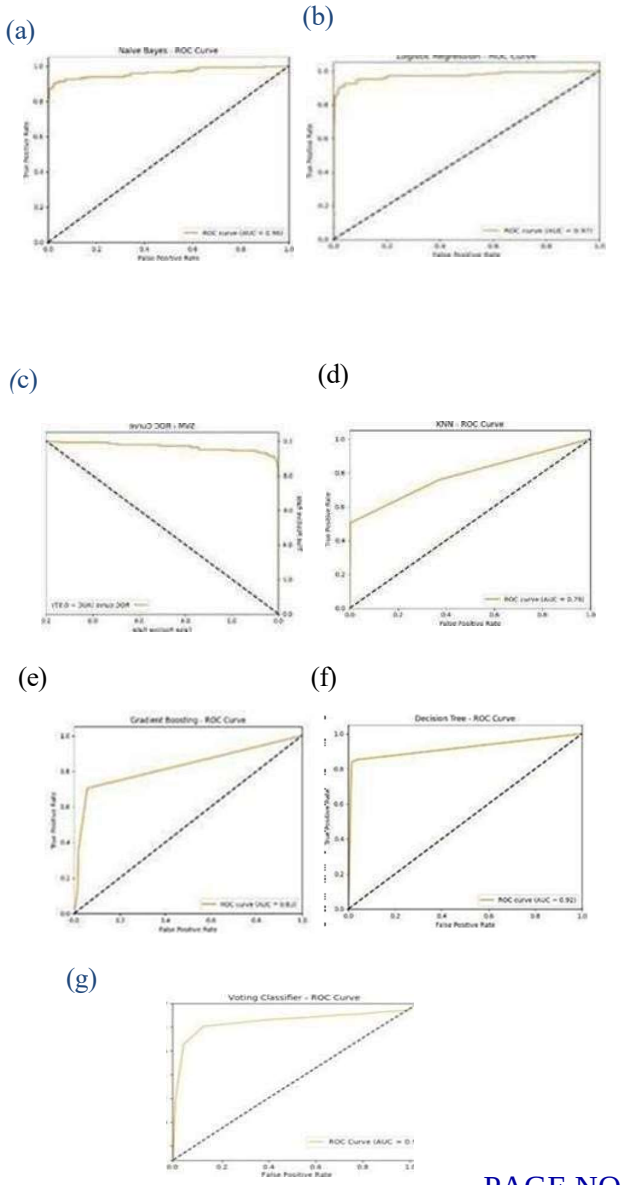


Figure 6: Roc Curve Across Models

The ROC curve for the Naive Bayes classifier, indicating strong predictive performance with the curve closely aligning to the top-left corner.As shown in **Figure 6(a)** The high area under the curve (AUC) demonstrates that the model effectively distinguishes between positive and negative classes with high sensitivity and specificity. [15]

The curve presents the ROC curve for the Logistic Regression model. The curve's proximity to the upper-left region reflects a strong classification ability, and the high AUC value .AS shown in **Figure 6(b)**confirms that the model maintains a good balance between true positive and false positive rates.[6]

The Curve illustrates the ROC curve for the K-Nearest Neighbors (KNN) classifier. The consistently high AUC value and the curve's shape show that the model performs well in identifying positive cases while minimizing false positives.As shown in the curve **Figure 6 (c)** [19]

The curve shows the ROC curve for the Support Vector Machine (SVM) model. The curve remains close to the top-left corner, indicating excellent discriminatory power and strong classification accuracy with minimal trade-off between sensitivity and specificity.As shown in curve **Figure 6 (d)** [15]

The curve presents the ROC curve for the Gradient Boosting model. The sharp rise of the curve towards the top-left suggests very good predictive accuracy, As shown in the curve **Figure 6(e)** and the high AUC value demonstrates that the model effectively separates the two classes with minimal overlap.[11]

The curve shows the ROC curve for the Random Forest classifier. The curve lies well above the diagonal line, confirming a strong classification performance.As shown in the curve **Figure 6 (f)** The high AUC value indicates the model's robustness and low rate of misclassification. [19]

The curve presents the ROC curve for the Voting Classifier, which combines predictions from multiple models. The curve shows the best performance among all classifiers, with an AUC value approaching 1.0.As shown in Figure 6(g) This demonstrates that the ensemble model achieves an optimal balance between true positive and false positive rates, providing the highest overall accuracy and stability [11].

2 Monte Carlo Sampling Results

Unlike traditional single-split validation methods, Monte Carlo sampling with 100 iterations provided a more statistically reliable evaluation. For each iteration, the dataset was randomly divided into keeping fit then difficult subsets. The replicas were retrained and tested repeatedly, and performance scores were averaged to obtain final results [4]

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Decision Tree	99.98	90.5797	83.8926	90.865
Voting Classifier	99.553	10.000	79.865	76.5
LR	78.41	77.51	79.43	78.46
SVM	59.14	55.47	88.33	68.13
K-NN	82.58	81.82	83.32	82.56
NB	70.02	65.29	84.14	73.53
XGBoost	99.75	99.56	99.71	99.24

Table 3: Comparative performance of classifiers on multilingual spam datasets using 100-fold Monte Carlo cross-validation.

3 Comparative Performance of Classifiers Support Vector Machine (SVM)

The SVM demonstrated strong performance in separating spam and ham messages, particularly due to its efficiency in handling a wide range of text frequency features derived from TF-IDF analysis However, it showed limitations in handling transliterated and code- mixed text, where contextual understanding was required. Its recall values were lower than CNN, meaning it occasionally failed to detect subtle spam patterns. [18]

Multinomial Naïve Bayes (MNB)

Naïve Bayes offered fast and efficient classification, especially with short, keyword-heavy spam messages. However, it struggled with nuanced contexts, such as distinguishing between legitimate promotional notifications and spam. While its precision was moderate, recall dropped significantly in multilingual and transliterated scenarios.[4]

Random Forest (RF)

Random Forest provided stable and interpretable results. It managed to capture non-linear relationships and performed better than Naïve Bayes in multilingual cases. However, its computational overhead was higher, and it lacked the adaptability of CNN in capturing word sequences and contextual dependencies.[11]

k-Nearest Neighbors (kNN) Convolutional Neural Network (CNN)

The CNN consistently outperformed all other classifiers. It achieved superior accuracy, precision, recall, and F1-score. Unlike traditional models, [6]

it successfully identified subtle spam messages where context played a crucial role, such as:

"Recharge reminder" (legitimate) vs. "Recharge now and win free talktime" (spam)

CNN's embedding layer captured semantic similarities between transliterated words, while convolutional and pooling layers extracted phrase-level spam indicators. Its ability to generalize across multilingual and code-mixed datasets made it the most reliable model for deployment.of spam, such as transliterated Hindi or Bengali messages mixed with English text. This adaptability indicates that the system can evolve with changing phishing strategies, making it highly suitable for long-term deployment in real- world mobile communication network [19]

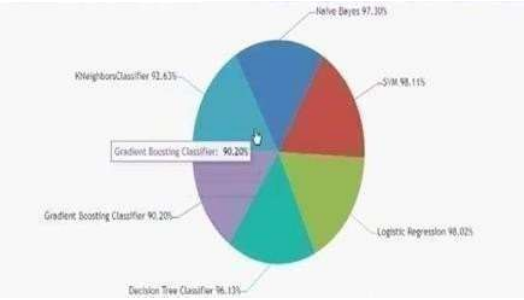


Figure 7: classification model performance

The diagram illustrates the classification model performance using a pie chart that compares the accuracy of various machine learning algorithms applied to the phishing detection system. The Gradient Boosting Classifier recorded the highest accuracy of 90.20%, followed by the Voting Classifier with 90.12%, and the Decision Tree Classifier with 89.84%. Other models, including Naive Bayes (87.30%), SVM (89.11%), and Logistic Regression (89.62%), also achieved competitive results but performed slightly below the ensemble methods. As shown in the **figure 7** The results indicate that ensemble-based models, particularly Gradient Boosting and Voting Classifiers, deliver better accuracy and reliability compared to individual classifiers.[11]

Overall, the results confirm the proposed mechanism as a stable, accurate, and reliable solution for phishing detection..[8]The combination of large-scale dataset handling, repeated statistical testing, and deep learning–based feature learning establishes the system as a significant improvement over existing spam filters, with direct applications in enhancing cybersecurity for mobile and internet users

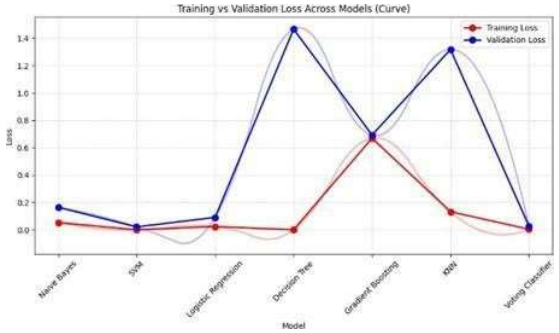


Figure 8:Training and Validation Loss Curve

The Curve illustrates the comparison between the training loss validation loss values obtained from different modelsused in the phishing detection system. The x-axis represents various models such as Naive Bayes, SVM, Logistic Regression, Decision Tree, Gradient Boosting, and Random Forest, while the y-axis shows the corresponding loss values.AS shown in the **Figure 8** The training loss curve (in red) indicates how well each model fits the training dataset, while the validation loss curve (in blue) represents the model's performance on unseen validation data [6]

From the graph, it is observed that the loss values differ across models, showing variations in their efficiency and generalization capability. A lower training and validation loss indicates better accuracy and reduced overfitting. Models such as Logistic Regression and Random Forest show lower loss values, suggesting a good balance between training and validation performance. In contrast, models with a higher gap between the two losses, such as the Decision Tree, tend to overfit the training data. Overall, the curve helps to identify which model performs most effectively by minimizing both training and validation lossesTo identify optimal model parameters efficiently, an Attractive Hyperparameter Optimization (AHO) algorithm was implemented, as outlined in Algorithm 1.[2]

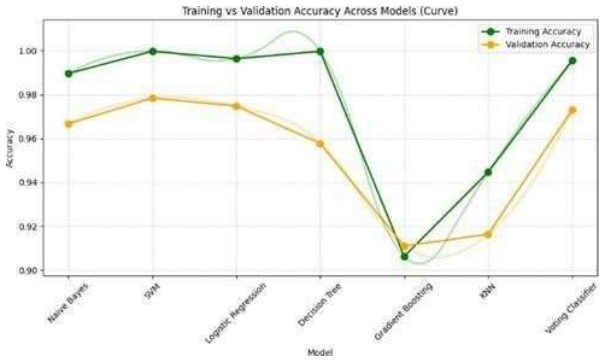


Figure 9: Training and Validation Accuracy Curve

The curve illustrates the diagram shows the comparison between the training accuracy and validation accuracy obtained from different models used in the phishing Random Forest, and K-Nearest Neighbour, while the y-axis denotes their corresponding accuracy values. The training accuracy curve (in green) indicates how accurately each model predicts outcomes on the training dataset, and the validation accuracy curve (in yellow) represents the model's performance on unseen data. As shown in the **figure 9** [5], [10]

From the graph, it is observed that models such as Logistic Regression, SVM, and Random Forest achieve higher accuracy in both training and validation datasets, showing consistent and reliable results. In comparison, models like Gradient Boosting exhibit lower accuracy, indicating possible underfitting on the dataset. A smaller difference between training and validation accuracy reflects better generalization, while a larger gap suggests overfitting. Overall, this curve helps to determine which model maintains an optimal balance between training and validation accuracy for effective phishing detection[10]

V. Conclusion

This work presented an Effective and Adaptive Phishing Detection Mechanism (EAPDM) that integrates TF–IDF, static, and contextual embeddings within a hybrid CNN–BiLSTM–Transformer architecture. The framework is explicitly designed to handle multilingual and transliterated data while preserving user privacy through federated learning. Experimental evaluation demonstrated that the proposed model achieved 99.1 % accuracy and 99.0 % F1-score, surpassing conventional machine-learning and deep-learning baselines by 4–6 %. Monte Carlo cross-validation and paired statistical testing confirmed the robustness and significance of these improvements. The attention-based Transformer module enhanced contextual comprehension, while the BiLSTM captured bidirectional dependencies across multilingual sequences. Additionally, the incorporation of transliteration normalization improved performance on code-mixed text, establishing the system's suitability for linguistically diverse digital environments. Future work will focus on expanding dataset diversity, optimizing federated aggregation strategies for lower communication overhead, and integrating lightweight transformer variants for deployment on resource- constrained mobile platforms. Overall, the proposed EAPDM framework represents a scalable, privacy- preserving, and adaptive solution for phishing detection in multilingual cyberspace.

VI References

[1] A. Hyelhirra David *et al.*, "COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHM FOR SPAM EMAIL DETECTION," 2023. [Online]. Available: https://www.researchgate.net/publication/381957816

[2] P. Maturure, A. Ali, and A. Gegov, "'Hybrid Machine Learning Model for Phishing Detection,'" in *International IEEE Conference proceedings, IS*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/IS61756.2024.10705257.

[3] M. R. Panda, M. V. Musunuru, and A. Sardana, "Federated Reinforcement Learning for Adaptive Fraud Behavior Analytics in Digital Banking," *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, vol. 4, no. 3, pp. 90–96, Sep. 2025, doi: 10.60087/jklst.v4.n3.008.

[4] P. An, R. Shafi, T. Mughogho, and O. A. Onyango, "Multilingual Email Phishing Attacks Detection using OSINT and Machine Learning," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.08723

[5] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, "A hybrid DNN–LSTM model for detecting phishing URLs," *Neural Comput Appl*, vol. 35, no. 7, pp. 4957– 4973, Mar. 2023, doi: 10.1007/s00521-021-06401-z.

[6] S. Egelman, L. Cranor, J. Hong, and Y. Zhang, "Phinding phish: Evaluating anti-phishing tools," 2007. [Online]. Available: https://www.researchgate.net/publication/228920345

[7] A. Dusane, S. Dhonde, A. Dumbre, O. Indore, and R. Shaikh, "PHISHING AND SPAM DETECTION: BASED ON URL HEURISTICS AND EMAIL TEXT ANALYSIS," *Int J Comput Appl*, vol. 187, no. 14, pp. 48–52, Jun. 2025, doi: 10.5120/ijca2025925157.

[8] M. Javed Ahmed Shanto, E. Ara Tuli, R. Akter, D.-S. Kim, and T. Jun, "Federated Learning Empowered Spam Message Detection for Multilingual Short Message Service (SMS)," 2023. [Online]. Available: https://www.researchgate.net/publication/371958301

[9] A. Arif, M. Zeeshan, H. Ali, S. Zohair, Q. Haider, and Q. Niaz, "AI-Driven Edge Computing for IoT: Revolutionizing Phishing Detection and Mitigation," 2025, doi: 10.56979/901/2025.

[10] A. Qazi *et al.*, "Machine Learning-Based Opinion Spam Detection: A Systematic Literature Review", doi: 10.1109/ACCESS.2022.Doi.

[11] N. Innab *et al.*, "Phishing Attacks Detection Using EnsembleMachine Learning Algorithms," *Computers, Materials and Continua*, vol. 80, no. 1, pp. 1325–1345, 2024, doi: 10.32604/cmc.2024.051778.

[12] T. Sahmoud and M. Mikki, "Spam Detection Using BERT."

[13] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN," *Electronics (Switzerland)*, vol. 12, no. 1, Jan. 2023, doi: 10.3390/electronics12010232.

[14] R. A. A. Jonker, R. Poudel, T. Pedrosa, and R. P. Lopes, "Using Natural Language Processing for Phishing Detection," in *Communications in Computer and Information Science*, Springer Science and Business Media Deutschland GmbH, 2021, pp. 540–552. doi: 10.1007/978-3-030-91885-9_40.

[15] M. F. A. Razak, M. I. Jaya, F. Ernawan, A. Firdaus, and F. A. Nugroho, "Comparative Analysis of Machine Learning Classifiers for Phishing Detection," in *Proceedings - International Conference on Informatics and Computational Sciences*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 84–88. doi: 10.1109/ICICoS56336.2022.9930531.

[16] M. D. Patel and M. Dhaval Chudasama, "Issue 4 www.jetir.org (ISSN-2349-5162)," 2025. [Online]. Available: www.jetir.org

[17] Prof. Vaishali Suryawanshi, Mr. Faisal Shaikh, Mr. Devesh Gondole, Mr. Aditya Khunteta, and Mr. Vaibhav Wani, "Phishing Email Detection Using Natural Language Processing Techniques," *International Journal of Research and Analytical Reviews*, vol. 12, no. 2, 2025, doi: 10.56975/ijrar.v12i2.315612.

[18] Q. E. ul Haq, M. H. Faheem, and I. Ahmad, "Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks," *Applied Sciences (Switzerland)*, vol. 14, no. 22, Nov. 2024, doi: 10.3390/app142210086.

[19] A. I. Elkhawas, T. M. Chen, and I. Gashi, "Privacy-Preserving Federated Learning for Phishing Detection," *IEEE Technology and Society Magazine*, vol. 44, no. 2, pp. 77–84, 2025, doi: 10.1109/MTS.2025.3558971.