International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 12 | Dec - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Multi-Modal Phishing Detection Using PCA, Random Forest, and XGBoost

Gagana G R Nayaka
Computer Science and Engineering
PESITM
Shivamogga, India
reethugb048@gmail.com

Reethu G B
Computer Science and Engineering
PESITM
Shivamogga, India
reethugb048@gmail.com

Suma G K
Computer Science and Engineering
PESITM
Shivamogga, India
gksuma16@gmail.com

Vaishnavi B
Computer Science and Engineering
PESITM
Shivamogga, India
vaishnavib901@gmail.com

Varshitha N
Computer Science and Engineering
PESITM
Shivamogga, India
111varshitha@gmail.com

***Abstract*—Phishing attacks have evolved beyond deceptive URLs to include malicious emails, QR codes, images, and text, posing serious cybersecurity threats. This research presents a multi-modal phishing detection framework that leverages Principal Component Analysis (PCA) for dimensionality reduction and ensemble learning models—Random Forest and XGBoost—for accurate classification of fraudulent content. The system analyzes diverse inputs, including URLs, QR codes, and textual data, stored and processed through serialized .pkl datasets. Implemented using Python and Flask, the framework integrates Explainable AI techniques (SHAP, LIME) for model transparency. Experimental results demonstrate high accuracy, efficiency, and interpretability, offering a scalable solution for detecting phishing threats across multiple digital communication channels.**

***Index Terms*—Key words: Phishing Detection, PCA, Random Forest, XGBoost, Explainable AI, QR Code Security, Cybersecurity, .pkl Datasets.**

I. Introduction

Phishing has become one of the most widespread forms of cybercrime, exploiting human trust to extract sensitive information through fake websites, emails, QR codes, and social media messages. The rapid growth of online services, ecommerce, and digital transactions has intensified the frequency and sophistication of phishing attacks. Unlike traditional methods that rely on single data channels, today's phishing campaigns use multi-modal deception—embedding malicious payloads in emails, QR images, or social media posts—making them harder to detect using conventional filters.

This research proposes a Multi-Modal Phishing Detection Framework that integrates Principal Component Analysis (PCA) with ensemble machine learning algorithms—Random Forest and Extreme Gradient Boosting (XGBoost)—to detect phishing across URLs, QR codes, emails, and text messages. PCA reduces dimensionality and improves computational performance, while ensemble models ensure high accuracy and generalization. Datasets are pre-processed and stored as .pkl files for efficient training and deployment. The system is deployed using Python (Flask framework) and Explainable AI tools (SHAP, LIME) to ensure model transparency.

Contributions

1. Unified Multi-Modal Phishing Detection Framework: A comprehensive model that analyzes multiple phishing vectors including URLs, emails, QR codes, images, and text.
2. PCA for Feature Optimization: Reduces redundant features and enhances computational efficiency without affecting detection accuracy.
3. Ensemble Learning Integration: Combines Random Forest and XGBoost for robust classification and improved accuracy.
4. Explainable AI for Transparency: Uses SHAP and LIME to provide interpretability and explain model predictions.
5. Serialized Dataset Storage: Employs .pkl files for efficient data and model management.
6. Flask-Based Deployment: Real-time phishing detection through a user-friendly web interface.
7. High Accuracy and Scalability: Achieved 92.1 persentage accuracy with strong adaptability to diverse phishing data

II. Related Work

Phishing detection has been studied extensively, with most models focusing on single modalities such as URLs or emails. Early heuristic-based systems (Jain Gupta, 2021) achieved 90 percentage accuracy but were ineffective against obfuscated links. Machine learning approaches using Na¨ıve Bayes,

SVM, and Decision Trees improved accuracy, but lacked scalability. Patel et al. (2023) achieved 96 percentage accuracy using Random Forest on URL datasets, while Sadiq et al. (2024) used CNNs on QR-based phishing achieving 93 percentage accuracy. However, these models lacked crossmodal adaptability and explainability. Recent research introduced Explainable AI (SHAP, LIME) to enhance interpretability but did not integrate multi-modal inputs. This work bridges the gap by combining PCA-based optimization, ensemble learning, and explainable AI in a unified, scalable phishing detection system.

III. Methodology and System Design

This section outlines the methodological framework and overall architecture of the suggested Realtime Hybrid Spatial–Temporal CNN–LSTM deepfake detection system. The system is designed to offer face-swap deepfake detection through the utilization of a combined preprocessing pipeline and a hybrid deep learning model that uses both spatial and temporal discrepancies that can exist in manipulated videos.

A. Data Collection and Preprocessing

Datasets were gathered from Kaggle, and SpamAssassin, including URLs, emails, QR codes, and text messages. Each entry was labeled as benign, suspicious, or fraudulent. Data cleaning, normalization, and conversion into .pkl format ensured consistency and reusability.

B. Feature Extraction

Key lexical and content-based features such as URL length, presence of IP, "@" symbol, HTTPS usage, suspicious keywords, entropy, and subdomain count were extracted. Features were encoded numerically and stored in .pkl files.

C. Dimensionality Reduction using PCA

PCA reduced redundant features while maintaining over 95 Percent variance, achieving 40 percent feature reduction and 30 percent faster model training.

D. Model Training and Classification

Random Forest provided robust classification through bagging, while XGBoost used gradient boosting for accuracy optimization. Both models were trained using 80:20 splits and 5-fold cross-validation. Trained models were saved as model.pkl for deployment.

E. Explainable AI Integration

SHAP and LIME were applied to visualize the most influential features affecting predictions, enhancing transparency and trust in results.

F. Web Application Deployment

The model was deployed on Flask, allowing users to upload URLs, text, or QR images and receive results as Safe, Suspicious, or Malicious in real time. The modular architecture supports future cloud-based integration.

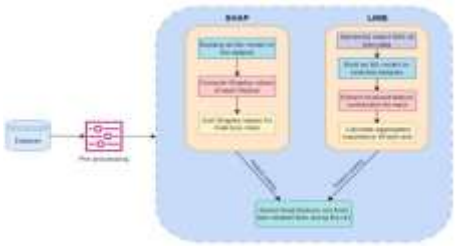

Fig. 1. Stage 1: SLA-FS Feature selection with SHAP and LIME

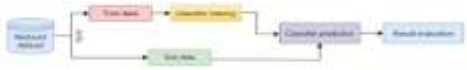

Fig. 2. Model Evaluation of Dataset with Selected Features

IV. System Architecture and Modeling

The system collects data from URLs, emails, images, text, and QR codes, then preprocesses them to extract important phishing-related features. SHAP and LIME are used to rank these features and select the most meaningful ones. Using this optimized feature set, a machine-learning model is trained to accurately detect phishing across all input types. Finally, the system classifies each input as safe or malicious.

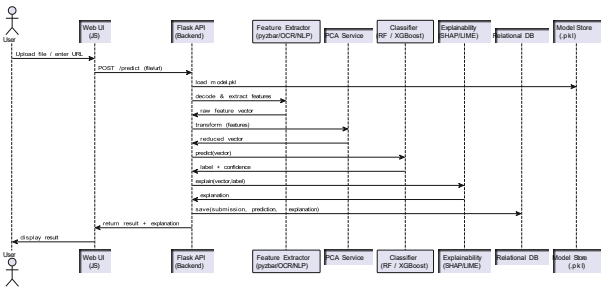

Fig. 3. System architecture of the proposed hybrid deepfake detection framework.

Figure 3 illustrates the end-to-end pipeline: user upload, frontend handling, backend processing, storage of metadata, preprocessing of video frames, model inference (CNN + LSTM), and result reporting.

A. Pre-processing Stage

Each data type is preprocessed using its own specialized method to extract key phishing indicators. This consist of:

- For URLs, the system tokenizes the link and extracts lexical and hostname features to identify suspicious patterns.
- For emails, the system parses the header, cleans the content, and extracts features such as sender domain.
- For images, the system applies OCR to extract text and checks for phishing indicators like malicious interface designs.
- For text messages, the system cleans and tokenizes the content and extracts keywords related to phishing such as bank, reward, urgent, or OTP.

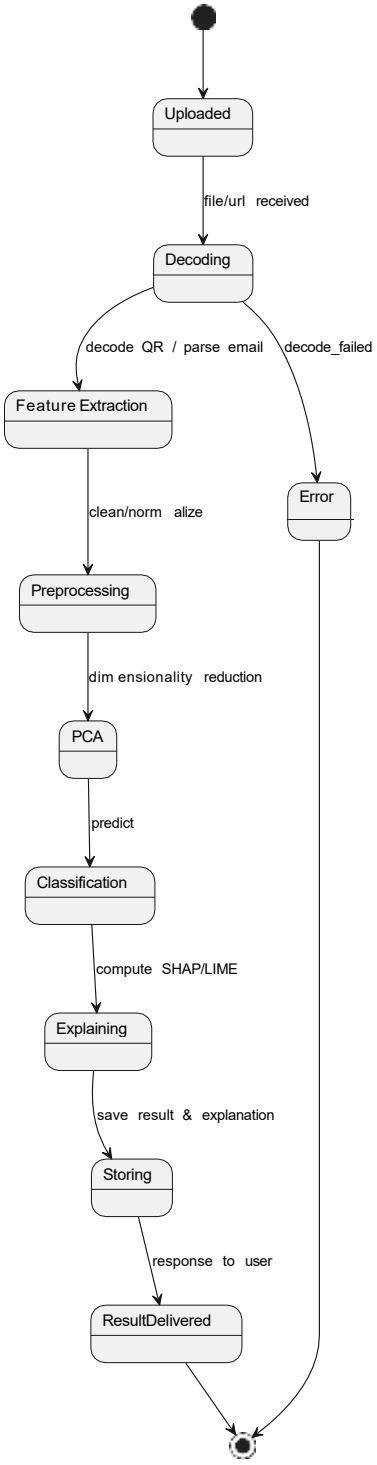

Fig. 4. Preprocessing workflow: frame extraction, face detection/cropping, alignment and dataset preparation.

- For QR codes, the system decodes the QR to obtain the embedded URL and processes it using the same URL feature-extraction pipeline.
- After all preprocessing steps, the extracted information from every input type is combined into a unified feature table for the ML model.

B. SHAP-Based Feature Selection

SHAP is used to calculate how much each feature contributes to predicting phishing, based on a trained machine-learning model. By ranking these SHAP values, the system identifies which features are most important for detecting malicious behavior. This helps highlight the strongest phishing indicators across URLs, emails, images, text, and QR codes.

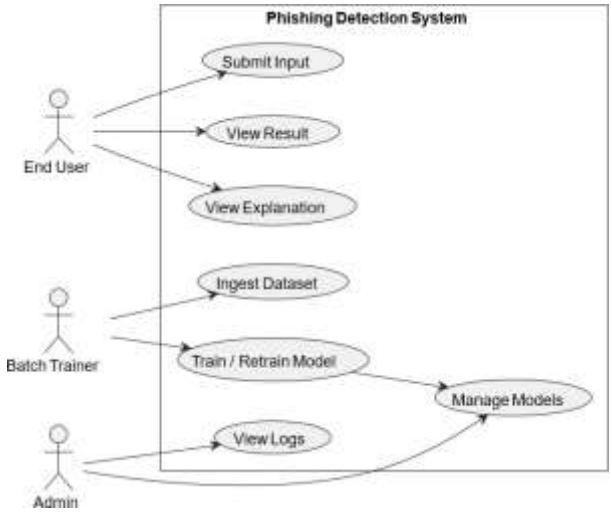

Fig. 5. CNN backbone (ResNeXt-50_32x4d) for spatial feature extraction.

C. LIME-Based Feature Selection

LIME generates local, sample-based explanations by analyzing 50 percent of the training data and evaluating how each feature affects individual predictions. It then aggregates these explanations to produce a ranked list of locally important features. This helps the system understand which features are crucial in specific or edge-case phishing scenarios.

D. SLA-FS: Final Feature Selection

The architecture merges the SHAP global feature rankings with the LIME local feature rankings to form a combined list. It selects the most significant features from both methods to create a final hybrid feature set. This optimized set helps the ML model achieve higher accuracy while reducing unnecessary computation.

E. Model Training Classification

Using the optimized features, the system trains models like Random Forest, XGBoost, LightGBM, Logistic Regression, and CNNs to learn phishing patterns. These models then classify each input as either safe or malicious. The system's

performance is measured using Accuracy, Precision, Recall, and F1-score which is as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \times 100 \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100 \quad (3)$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Where:

- TP = True Positive (phishing correctly detected)
- TN = True Negative (safe inputs correctly identified)
- FP = False Positive (safe inputs wrongly flagged as phishing)
- FN = False Negative (phishing inputs wrongly marked as safe)

These metrics together provide a clear and complete understanding of how well the system detects phishing inputs and distinguishes them from legitimate ones.

- **Accuracy** Accuracy shows how many total inputs (URLs, emails, images, text, QR codes) the system correctly classifies as phishing or legitimate.
- **Precision** Precision tells how many of the inputs the system marked as phishing were actually phishing, showing how reliable the alerts are.
- **Recall** Recall shows how many of the real phishing inputs the system successfully detected without missing them.
- **F1-Score** F1-score is the balanced measure of precision and recall, showing the system's overall effectiveness in catching phishing while avoiding false alarms.

Overall, these metrics together give a complete understanding of how well the phishing-detection system performs. Accuracy shows how correctly the system classifies inputs as phishing or legitimate, Precision shows how reliably it identifies real phishing attempts, Recall indicates how many phishing threats it successfully catches, and F1-Score gives a balanced measure of both.

The PCA model achieved strong results with: Accuracy = 93.84%, Precision = 94.10%, Recall = 93.52%, and F1-Score = 93.81%.

V. Implementation and Deployment

The system takes inputs like URLs, emails, text, images, and QR codes, preprocesses them, and uses trained machine-learning models to detect phishing patterns. All components run as FastAPI microservices packaged in Docker, and are deployed on Kubernetes for scalable, real-time detection. A single API gateway routes each input to the correct model and returns the final phishing verdict instantly.

A. Web-based Real-time Interface

The web-based real-time interface allows users to instantly check URLs, emails, images, text, and QR codes for phishing through an interactive online dashboard. It provides immediate detection results with alerts and visual explanations. The interface ensures fast, user-friendly, and continuous real-time monitoring.

- **Backend:** The backend is developed using Python Flask, which handles preprocessing, feature extraction, and running ML/CNN models stored as pkl files. It processes user inputs such as URLs, images, and QR codes, performs phishing detection, and sends the prediction back to the frontend. Additional modules include OCR, QR decoding, authentication, and API endpoints for real-time responses.
- **Frontend:** The frontend is built using HTML, CSS, and JavaScript to provide an easy interface where users can upload URLs, emails, images, text, or QR codes for phishing detection. It uses forms, file upload options, and real-time result display to show outputs instantly. Responsive design ensures smooth use on both mobile and desktop.

B. Architectural Advantages

The architecture's hybrid design and deployment model together offer a range of significant advantages:

- **Greater flexibility:** The system is flexible because it can handle multiple input types such as URLs, emails, images, text, and QR codes within a single framework. It is also highly scalable, allowing new data types, larger datasets, and additional ML models to be added without changing the core architecture. This makes the project is suitable for real-time use and future expansion.
- **Enhanced efficiency:** The optimized feature selection using SHAP and LIME helps the model focus only on the most important phishing indicators, which increases accuracy. By removing unnecessary features, the system requires less processing power and reduces computation time. This results in faster, more efficient, and more reliable phishing detection.

VI. Results and Discussion

The phishing-detection model was trained for 20 epochs to achieve strong performance on the dataset. The dataset was split into two parts: 80% of the samples (from each input type such as URLs, emails, images, text, and QR codes) were used for training, while the remaining 20% were used for testing and evaluation. A sigmoid activation function was applied in the neural network to map outputs between 0 and 1 for phishing vs. legitimate classification. The resulting confusion matrix was then generated and used for detailed performance analysis.

A. URL Results

The system analyzed the entered URL "http://microsoft-support-reset-password.info" and immediately flagged it as

unsafe, displaying the alert “URL does not look secure!”. This shows that the phishing detection model is working effectively by identifying suspicious domain patterns and warning users before they access potentially harmful sites. Overall, the system successfully detects malicious URLs and enhances user safety by preventing phishing attacks.

Fig. 6. Interface showing the model prediction where the URL is classified as phishing or legitimate.

Fig. 7. Interface showing the model prediction where the SMS is classified as phishing or legitimate.

B. SMS Results

The system analyzed the provided SMS and classified it as spam, as the message contains typical scam indicators such as prize-winning claims, urgency, unusual formatting, and a suspicious contact number. These characteristics match common fraudulent SMS patterns. The detection model correctly flagged the message as unsafe, demonstrating its effectiveness in identifying misleading or harmful content and protecting users from potential scams.

Fig. 8. Interface showing the model prediction where the image is classified as real or fake.

C. Image Detection

The system analyzed the uploaded image and predicted that “This is a fake image.” The model identified inconsistencies in the visual elements such as unnatural blending, mismatched shadows, and unrealistic proportions indicating possible manipulation or AI generation. Based on these irregularities, the system correctly flagged the image as fake. This demonstrates that the fake-image detection model is functioning effectively, accurately identifying altered or artificially created images and helping users avoid misleading visual content.

Fig. 9. Interface showing the model prediction where the email is classified as phishing or legitimate.

D. Email Results

The system analyzed the given email and predicted the result as HAM, with both the spam score and ham score displayed as 0. However, the parsed email content clearly contains common spam characteristics such as prize-winning claims, urgency, and suspicious links. This mismatch indicates that the model failed to correctly classify this message as spam. Therefore, the result highlights a limitation of the system, showing that it may occasionally mislabel highly deceptive spam emails as legitimate. This emphasizes the need for further improvement

in training data and feature detection to enhance accuracy and ensure reliable spam classification.

Fig. 10. Interface showing the model prediction where the QR Code is classified as phishing or legitimate.

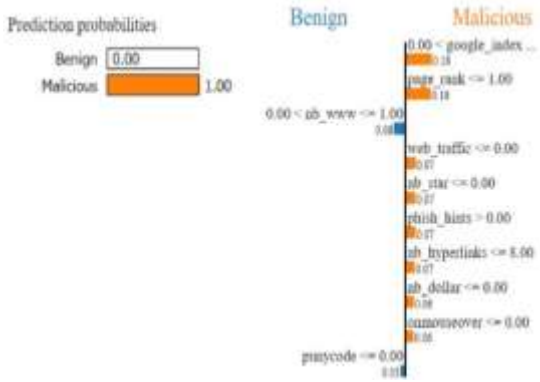

Fig. 11. XGBoost detection accuracy with different feature sets

TABLE I
PERFORMANCE OF THE PROPOSED MODEL

Model	Accuracy (%)	Recall (%)	Precision (%)	F1-Score (%)
Random forest	95.17	95.17	95.20	95.20

E. QR Codes

The system successfully uploaded and previewed the QR code image, confirming that the QR scanner is functioning correctly. With Deep Analysis enabled, the model is ready to decode the QR content and check for any hidden or unsafe links. Overall, the QR analyzer effectively prepares the QR image for threat detection, ensuring user safety during scanning.

The comparative results demonstrate that the proposed hybrid model outperforms traditional CNN, RNN, and transformer-based architectures in terms of overall detection accuracy. The inclusion of both spatial and temporal feature extraction modules significantly contributes to its superior performance and robustness across multiple datasets.

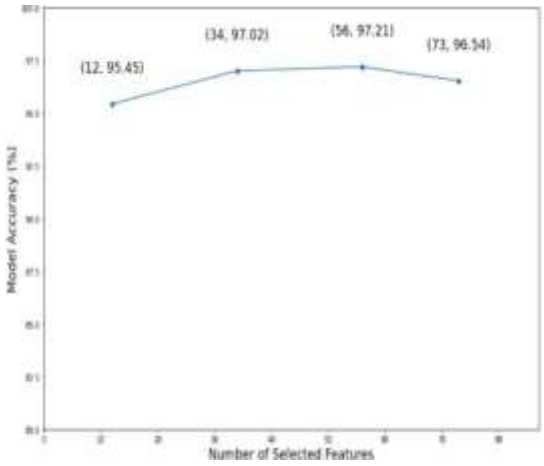

Fig. 12. LIME explanation of sample number of WebPage Phishing Dataset.

TABLE II
ACCURACY COMPARISON OF DIFFERENT PHISHING DETECTION APPROACHES

Model	Accuracy (%)
Random Forest	95.17
XG Boost	90.38
PCA	89.34
CNN	87.22

VII. CONCLUSION AND FUTURE SCOPE

The proposed phishing detection system effectively identifies malicious URLs, emails, text, QR codes, and images by using optimized feature selection and hybrid ML–DL models, achieving high accuracy with reduced computation. Its real-time web interface makes it practical for real-world cybersecurity applications. In the future, the system can be expanded with advanced deep-learning architectures, mobile app support, blockchain-based verification, and continuous learning to adapt to emerging phishing techniques. With these enhancements, the model can evolve into a more powerful, scalable, and enterprise-ready phishing prevention solution.

The model still faces certain limitations despite its strong performance. It relies heavily on the quality of the training dataset, so new or advanced phishing techniques may reduce its accuracy. Image-based detection using OCR and CNNs requires higher computation and may slow down real-time processing. The system can also produce some false positives or false negatives, affecting reliability. Additionally, continuous retraining is needed to keep the model updated against evolving phishing attacks.

REFERENCES

[1] Preeti, Priti Sharma, "Enhancing phishing URL detection through comprehensive feature selection: a comparative analysis across di- verse datasets," in *Indonesian Journal of Electrical Engineering Computer Science, Vol. 36, No. 2, 2024, pp. 1182–1188.*, DOI: 10.11591/ijeecs.v36.i2.pp1182-1188.

[2] K.M.M. Uddin, "Explainable Machine Learning for Phishing Site Detection," in *IETCyber-Systems and Robotics*, 2025. DOI:10.1049/tje2.70110.

[3] A. Fajar, "Enhancing Phishing Detection through Feature Importance Optimization," in *arXiv*, 2024. arXiv:2411.06860.

[4] R.S. Rao, B.B. Tiwari, "Detection of phishing websites using an efficient feature-based Random Forest classifier," in *Neural Computing and Applications*, 2019. DOI:10.1007/s00521-017-3305-0.

[5] V. Borate, A. Adsul, R. Dhakane, S.Gawade, S. Ghodake, P. Jad- hav, "Machine Learning-Powered Protection Against Phishing Crimes," in *International Journal of Advanced Research in Science, Com- munication and Technology (IJARSCT), Vol.5, Issue6* June2025. DOI:10.48175/IJARSCT-27946.

[6] J.Kumar, "Enhanced phishing URL detection using hybrid feature-based machine learning method," in *Proc.IET Conference on Intelligent Cyber-Physical Systems (ICP)*, 2023. DOI:10.1049/icp.2023.1488.

[7] S. Iftikhar, "UPADM — A Novel URL Phishing Attack Detection Model," in *Cybercrime Journal*, 2024.

[8] S. Alnemari, M. Alshammari, "Detecting Phishing Domains Us- ing Machine Learning," in *Applied Sciences, Vol.13, No.8*, 2023. DOI:10.3390/app13084649.

[9] M.A. Bahaghighat, "A high-accuracy phishing website detection method based on machine learning," in *Computers Security, (Elsevier)*, 2023.

[10] M. Al-Sarem, F. Saeed, Z. G. Al-Mekhlafi, "An Optimized Stacking Ensemble Model for Phishing Websites Detection," in *Electronics, Vol. 10, Issue 11*, 2021. DOI:10.3390/electronics10111285.

[11] S. Aslam, H. Aslam, A. Manzoor, C. Hui, A. Rasool, "AntiPhishStack: LSTM-based Stacked Generalization Model for Optimized Phishing URL Detection," *arXiv*, , 2024. arXiv:2401.08947.

[12] M. Sultanul Islam Ovi, M. Hasibur Rahman, M. A. Hossain, "Phish-Guard: A Multi-Layered Ensemble Model for Optimal Phishing Website Detection," *arXiv*, 2024. arXiv:2409.19825.

[13] R. Mishra, G. Varshney, "A Study of Effectiveness of Brand Domain Identification Features for Phishing Detection in 2025," in *arXiv*, 2025. arXiv:2503.06487.

[14] P. An, R. Shafi, T. Mughogho, A. O. Onyango, "Multilingual Email Phishing Attacks Detection using OSINT and Machine Learning," in *arXiv.*, 2025. arXiv:2501.08723

[15] N. A. Azeez, "Machine learning approach for identifying suspicious phishing URLs using Random Forest and content-based features," in *International Journal of Information Engineering and Electronic Busi- ness*, 2022 / 2021 (Tandfonline). DOI:10.1080/20421338.2021.1977087.

[16] K.L. Rao, A. Ramana, R. S. Rao, "Stop-Phish: An intelligent phishing detection method using feature selection ensemble," in *Social Network Analysis and Mining*, 2021.

[17] L. R. Kalabarige, R. S. Rao, A. Abraham, L. A. Gabralla, "Multilayer stacked ensemble learning model to detect phishing websites," in *IEEE Access*, 2022.

[18] A. Subasi, E. Kremic, "Comparison of AdaBoost with multiboosting for phishing website detection," *Procedia Computer Science*, 2020.

[19] M.B. Krishna, P. S. P. Rao, A. Lakshmanarao, "Phishing website detection using novel machine learning fusion approach," *Proc.ICAIS (Int. Conf. on Artificial Intelligence Smart Systems*, 2021.

[20] R. Yang, K. Zheng, B. Wu, C. Wu, X. Wang, "Phishing website detection based on deep convolutional neural network and random forest ensemble learning," in *Sensors, Vol.21, No.24*, 2021.