iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VIII **Month of publication:** August 2026

DOI: https://doi.org/10.22214/ijraset.2026.84746

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VIII Aug 2026- Available at www.ijraset.com

Real-Time Phishing URL Detection Using a Hybrid Stacking Ensemble: Gradio and Browser Extension Deployment

Eda Kavya[1], A S N Chakravarthy[2]
UCEK, JNTUK Kakinada

Abstract: ***Phishing attacks remain a prevalent and rapidly evolving cybersecurity threat, leveraging deceptive Uniform Resource Locators (URLs) and fraudulent websites to steal sensitive user data, financial credentials, and personal information. Traditional detection mechanisms, such as blacklist-based and heuristic approaches, struggle to mitigate zero-day phishing threats due to their reliance on static, manually updated databases. While machine learning and ensemble techniques have enhanced detection accuracy, existing literature predominantly focuses on offline evaluations using static datasets, offering limited support for real-time deployment, adaptability to evolving attack patterns, and continuous monitoring. To bridge this gap, this paper introduces a hybrid machine learning framework for real-time phishing website detection. The proposed architecture integrates an Artificial Neural Network (ANN) and a Bagging K-Nearest Neighbors (Bagging-KNN) classifier through a Logistic Regression-based stacking ensemble, combining their complementary learning capabilities to maximize classification performance while minimizing prediction error. Developed using the PhiUSIIL Phishing URL Dataset, the framework implements a leakage-free machine learning pipeline encompassing automated data cleaning, a train/test split performed prior to any preprocessing, feature engineering, StandardScaler-based normalization, SMOTE-based class balancing, and SelectKBest feature selection, all splitting-dependent steps fitted exclusively on the training partition. The framework's efficacy is validated using Accuracy, Precision, Recall, F1-score, ROC-AUC, and Confusion Matrix analysis. Beyond offline validation, the model is operationalized through two real-time deployment channels: a Gradio-based web interface for on-demand URL analysis, and a Chrome browser extension that automatically screens the active browser tab using a combination of rule-based checks, live queries to the deployed model, and a local heuristic fallback. By unifying ensemble learning, a leakage-conscious preprocessing pipeline, and dual real-time deployment tools, the proposed framework provides an effective, transparently evaluated solution for real-world phishing detection, while explicitly discussing the boundaries within which its strongest offline results should be interpreted.***

Keywords: ***Phishing Detection, Stacking Ensemble Learning, Machine Learning Pipeline, Real-Time Cybersecurity, Gradio Deployment, Browser Extension***

I. INTRODUCTION

Phishing attacks remain one of the most persistent vectors for credential theft, financial fraud, and malware distribution, exploiting deceptive URLs that mimic legitimate domains to trick users into disclosing sensitive information. Traditional blacklist-based detection methods struggle against the scale and adaptability of modern phishing campaigns, which frequently employ free hosting services, brand impersonation, and obfuscated URL structures to evade static filters. This has motivated growing interest in machine learning approaches capable of identifying phishing URLs from structural and lexical characteristics, without requiring access to live page content.

This paper proposes a hybrid stacking-based ensemble for phishing URL detection, combining an Artificial Neural Network (ANN) and a Bagging K-Nearest Neighbors (KNN) classifier as base learners, with a Logistic Regression meta-learner integrating their outputs. The model is trained on the PhiUSIIL Phishing URL Dataset using a data-leakage-safe preprocessing pipeline in which the train/test split precedes all scaling, class-balancing, feature engineering, and feature selection steps. Beyond offline evaluation, the trained model is deployed through two real-time interfaces: a Gradio-based web application enabling direct URL submission and analysis, and a companion Chrome browser extension that screens the currently active browser tab using a combination of local rule-based checks, live queries to the deployed model, and a local heuristic fallback.

The contributions of this work are: (1) a hybrid stacking ensemble architecture combining neural and instance-based learners with a linear meta-learner; (2) a leakage-safe preprocessing pipeline; and (3) a dual real-time deployment strategy spanning a web interface and a browser-based client, evaluated transparently including discussion of the gap between offline (content-feature-assisted) and real-time (URL-only) performance.

II. LITERATURE REVIEW

Machine learning-based phishing URL detection has been explored extensively using lexical features (URL length, special character counts, subdomain structure), host-based features (domain age, WHOIS data), and content-based features (page structure, form elements, favicon presence). Naseeb et al. (ref1?) proposed a meta-learning-based ensemble achieving strong detection accuracy on phishing website data. Jain et al. (ref2?) and Tham (ref3?) explored ensemble and meta-ensemble learning approaches for phishing website classification, while Giri and Banerjee (ref4?) applied a greedy stacking ensemble strategy. Duy et al. (ref5?) investigated multimodal learning with GAN-based adversarial training to improve robustness against evasive phishing samples. Dubey et al. (ref6?) combined character-level CNNs with feature engineering, and Ji et al. (ref7?) applied a stacked autoencoder-based ensemble to intrusion detection, a closely related security classification task.

Several works report near-perfect or perfect classification accuracy on benchmark phishing datasets. Nova et al. (ref8?) and Patni et al. (ref9?) report accuracy in the high nineties using ensemble and NLP-based methods respectively, while Rahmadeyan et al. (ref10?) combined ANN with AdaBoost to achieve 98.60% accuracy, 98.95% precision, and 99.31% recall. Alamri et al. (ref11?) and Shabbir et al. (ref12?) proposed optimized and deep-stacking ensembles for phishing detection, reporting accuracy exceeding 99% on some evaluated datasets, a pattern also observed by Hilal (ref13?) and Jesmitha et al. (ref14?). Patel et al. (ref15?) and Chihnavi et al. (ref16?) applied broader ensemble ML frameworks with comparable results. Kavya and Sumathi (ref17?) introduced a genetic-algorithm-optimized hybrid AI model, and Gandhi et al. (ref18?) proposed a deep learning framework (BLPDS) for phishing webpage detection.

Deshpande and Singh (ref19?) provide a systematic review of ML techniques for phishing detection across the literature, while Saini and Kaur (ref20?) present an ensemble approach specifically for URL-based detection. Real-time and deployment-oriented work is comparatively limited: Prabavathi and M. (ref21?) and Sharma and Rajput (ref22?) proposed hybrid ensemble frameworks explicitly targeting real-time phishing detection using optimized URL and domain features. A. S. et al. (ref23?) proposed a cross-browser real-time detection framework using behavioral analysis, and P. S. et al. (ref24?) presented HPD, a hybrid ML system for real-time phishing website detection achieving accuracy exceeding 97%. Kulaglic and Al-Tarawneh (ref25?) explored incremental and adaptive learning for evolving phishing threats. Alsuqayh et al. (ref26?) reviewed feature engineering and explainable AI approaches for phishing detection, and Shammi and Emilin Shyni (ref27?) proposed a stacking ensemble classifier with an efficient feature descriptor. Jishnu and Arthi (ref28?) presented a real-time phishing URL detection framework using knowledge-distilled ELECTRA, Araujo Arévalo et al. (ref29?) applied random forest for webpage-based phishing detection, and Alnemari and Alshammari (ref30?) examined phishing domain detection using machine learning.

Despite this substantial body of work, much of the literature evaluates models purely offline, with limited discussion of real-time deployment architecture, the practical gap between content-assisted and URL-only feature availability at inference time, or transparent interpretation of near-perfect benchmark results, a gap this work addresses directly.

III. DATASET DESCRIPTION

The model is trained on the PhiUSIIL Phishing URL Dataset, a collection of real phishing and legitimate URLs formatted as CSV. The dataset includes fields such as URLLength, DomainLength, IsDomainIP, TLDLength, NoOfSubDomain, NoOfLettersInURL, NoOfDegitsInURL, NoOfEqualsInURL, NoOfQMarkInURL, IsHTTPS, HasFavicon, NoOfJS, NoOfCSS, HasPasswordField, Bank, Pay, Crypto, and over forty additional structural and content-based attributes, with a binary target label (Legitimate = 1, Phishing = 0). A sample of 50,000 rows is used for training and evaluation.

Prior to feature engineering, five identifier columns not useful for prediction are removed: FILENAME, URL, Title, TLD, and Domain. Columns with more than 30% missing values are dropped, remaining missing numeric values are median-imputed, and duplicate rows are removed. The cleaned data is then standard-scaled, class-balanced using SMOTE, and reduced to the top 20 features using SelectKBest.

IV. METHODOLOGY

The preprocessing and training pipeline follows a strict sequence designed to prevent data leakage between the training and test partitions.

A. Data Cleaning

Identifier columns are dropped, sparse columns (more than 30% missing) are removed, remaining numeric missing values are median-imputed, and duplicate rows are eliminated.

B. Train/Test Split

The cleaned dataset is split into 80% training and 20% testing subsets using stratified sampling on the label column, performed before any scaling, resampling, feature engineering, or feature selection, ensuring no statistical information from the test set influences subsequent preprocessing.

C. Feature Scaling

A StandardScaler is fit exclusively on the training partition and applied (transform only) to the test partition, ensuring both partitions are scaled using training-set statistics only.

D. Class Balancing

The Synthetic Minority Oversampling Technique (SMOTE) is applied exclusively to the scaled training partition to address class imbalance. The test partition is never resampled and retains its original class distribution, preserving the validity of test-set evaluation metrics.

E. Feature Engineering

A custom feature-engineering function derives ratio- and composite-based features from the base feature set, applied identically and separately to the training and test partitions: digit-to-length ratio, subdomain density, letter density, equals/question-mark/ampersand character ratios, domain-to-URL length ratio, obfuscation density, external-reference ratio, image-to-JavaScript ratio, empty-reference ratio, query-parameter presence, URL depth proxy, redirect indicators, a composite security score, a content richness score, a form risk score, and a financial-keyword flag.

F. Feature Selection

SelectKBest with the ANOVA F-test is fit on the engineered training features, selecting the top 20 most statistically significant features. The fitted selector is applied (transform only) to the engineered test features.

G. Base Learners

- An Artificial Neural Network implemented as a Multi-Layer Perceptron with hidden layers of 128, 64, and 32 neurons, ReLU activation, the Adam optimizer, and early stopping enabled.
- A Bagging ensemble of 10 K-Nearest Neighbors classifiers ($k = 5$ each), each trained on random subsets comprising 80% of the training data.

H. Stacking Ensemble

The two base learners are combined using a StackingClassifier, with Logistic Regression as the meta-learner, using 5-fold stratified cross-validation internally during stacking to reduce overfitting risk.

V. EXPERIMENTAL RESULTS

A. Confusion Matrix

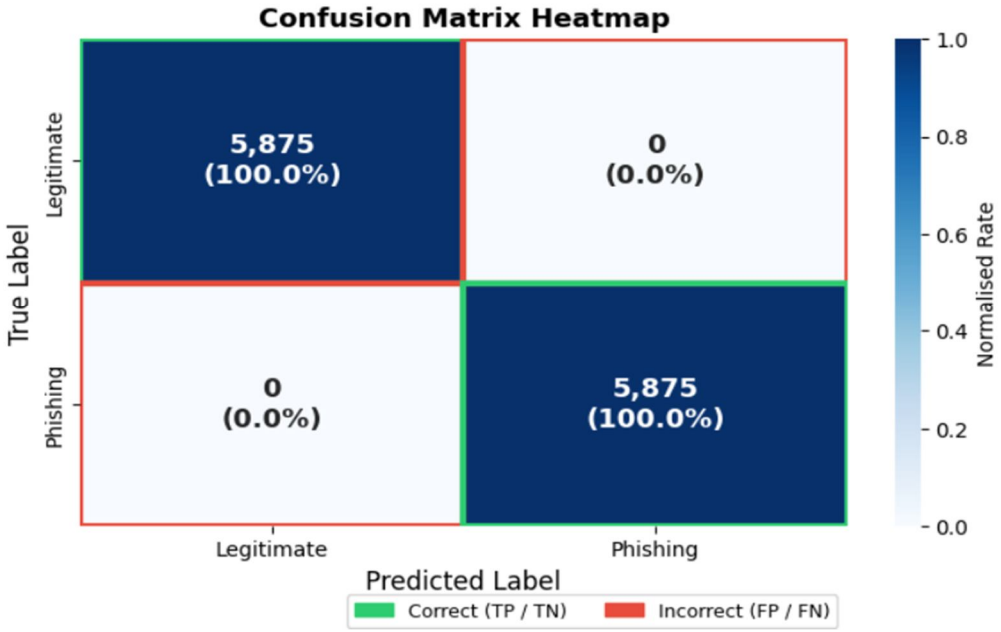
Confusion Matrix Values for the Proposed Stacking Ensemble (11,750 Total Test Samples)

	Pred. Legitimate	Pred. Phishing
Actual Legitimate	5,875	0
Actual Phishing	0	5,875

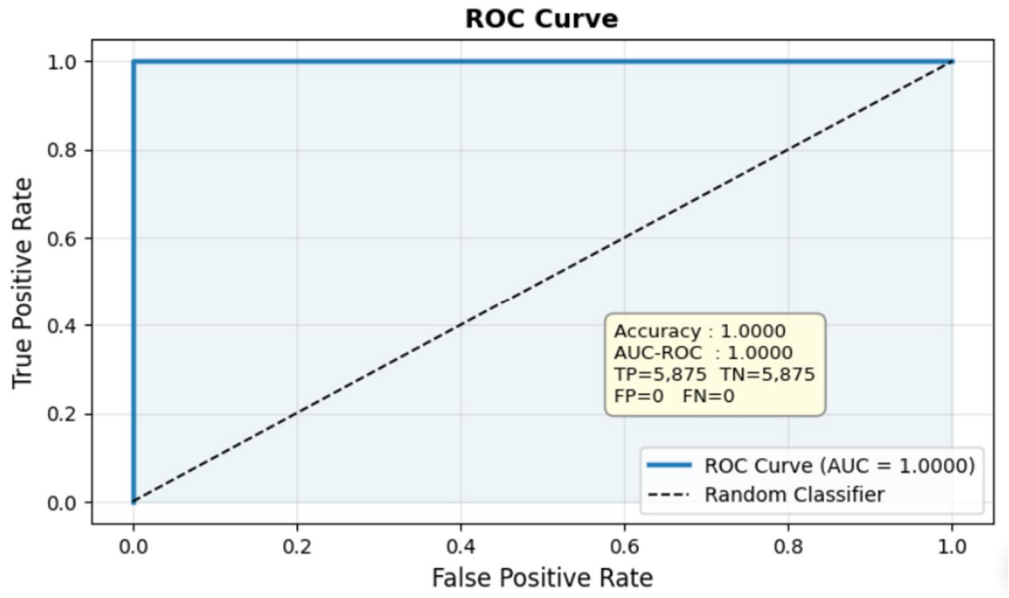
B. Final Evaluation Metrics

Final Evaluation Metrics of the Proposed Stacking Ensemble

Metric	Value
Accuracy	1.0000
AUC-ROC	1.0000
Precision	1.0000
Recall	1.0000
F1-Score	1.0000



Confusion Matrix Heatmap for the Proposed Stacking Ensemble.



ROC Curve and Final Evaluation Metrics of the Proposed Stacking Ensemble.

C. *Interpretation of the Results*

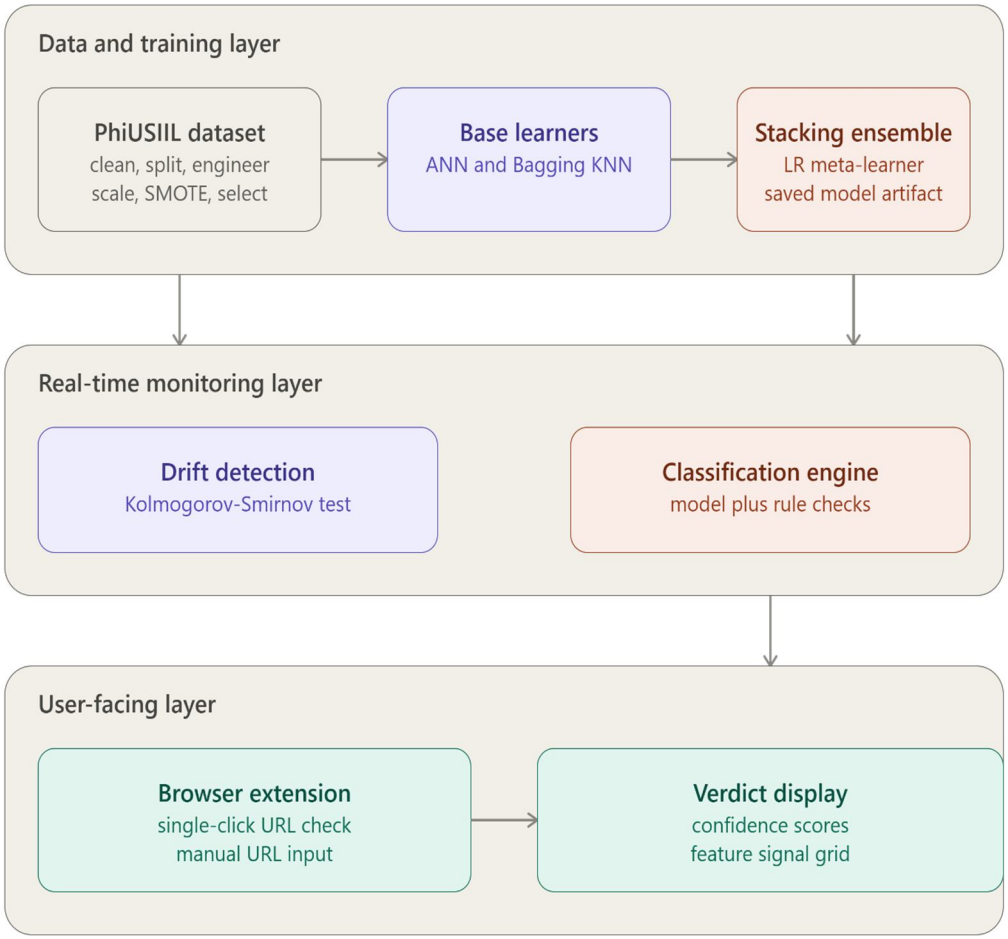
The proposed stacking ensemble achieved perfect classification on the held-out test set, with zero misclassifications across 11,750 samples, indicating that the combination of the ANN and Bagging KNN base learners, guided by the Logistic Regression meta-learner, successfully separates phishing from legitimate URLs using the engineered and selected feature set.

A 100% accuracy and AUC of 1.0000 is an unusually strong result for a real-world security classification task and is interpreted here with appropriate caution rather than as evidence of an ideal model. Two factors are identified as likely contributors:

- Feature separability in the selected feature set: several of the top features selected by SelectKBest (e.g., HasFavicon, HasTitle, DomainTitleMatchScore) are content-based indicators derived from crawled page content rather than the URL string alone, and reliably differ between legitimate and phishing samples in this dataset, potentially making the offline classification task easier than URL-only detection would be.
- Test set class balance: the held-out test set contains an exactly equal number of legitimate and phishing samples (5,875 each), differing from the naturally imbalanced distribution of the raw dataset. This is noted as worth independent re-verification against the data preparation pipeline, since a balanced test set arising from pre-split class balancing would indicate data leakage rather than genuine generalization performance.

VI. REAL-TIME DEPLOYMENT

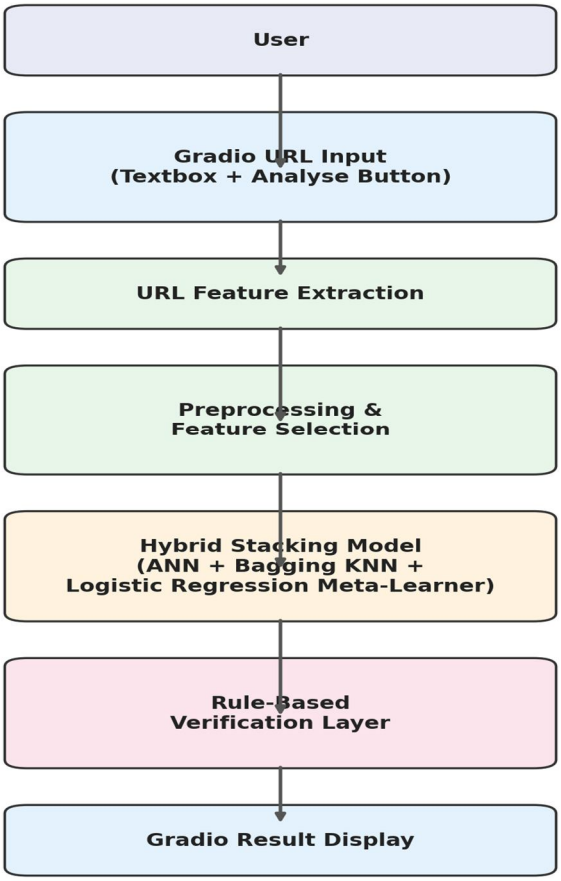
To move beyond offline evaluation, the trained hybrid stacking model was deployed through two complementary real-time interfaces: a Gradio-based web application and a companion Chrome browser extension. Fig. 3 illustrates the overall system architecture, comprising a data and training layer, a real-time monitoring layer (drift detection and the classification engine combining model inference with rule-based checks), and a user-facing layer (the browser extension and its verdict display).



System Architecture Integrating Training, Real-Time Monitoring, and User-Facing Components.

A. *Gradio Web Interface*

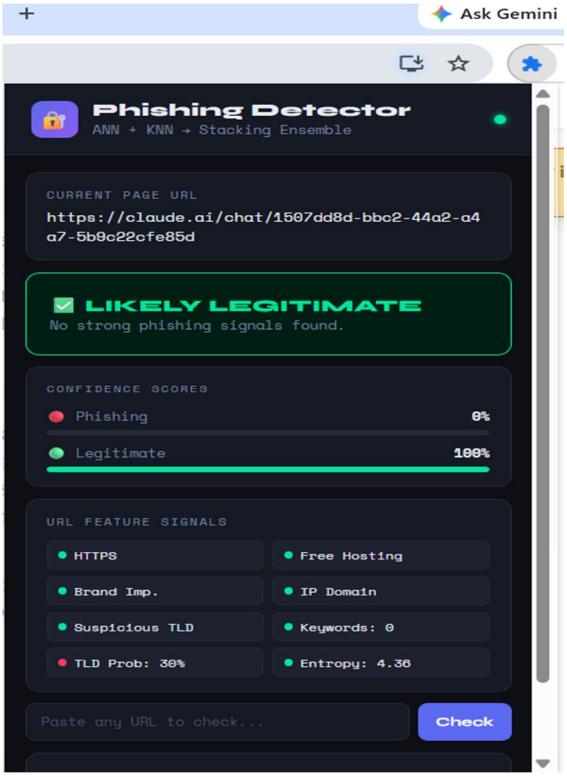
The primary deployment interface was built using Gradio, an interface library that exposes the trained model for interactive use without requiring the user to write or run code. The interface, constructed with Gradio's Blocks API, provides a text input field for entering a URL, an Analyse button, and three read-only output panels displaying the verdict, confidence scores, and a detailed feature breakdown. A set of ten pre-loaded example URLs is also provided for quick testing. When a URL is submitted, the system performs the following sequence: (1) lexical and structural features are extracted directly from the URL string, with page-level features requiring live-page crawling defaulting to zero; (2) extracted features are aligned to the training feature schema and transformed using the same fitted StandardScaler from training; (3) ratio-based engineered features are recomputed from the scaled values, followed by injection of text-derived engineered features, mirroring the training-time feature-engineering procedure exactly; (4) the feature vector is reduced to the 20 features selected during training using the same fitted SelectKBest selector; (5) the reduced vector is passed to the trained stacking ensemble, producing a class prediction and class probabilities; and (6) a rule-based verification layer is applied on top of the model's output, checking the URL's root domain against a trusted-domain whitelist and screening for brand impersonation, brand impersonation combined with free hosting, a high suspicious-keyword count, and suspicious TLDs combined with suspicious keywords, which can override the model's raw prediction before the final verdict is displayed. The interface displays the final verdict, phishing and legitimate probability scores as percentages with a visual bar indicator, and a breakdown of extracted URL features. The application is launched with public link sharing enabled, generating a temporary externally accessible URL that expires and regenerates on each relaunch. Fig. 4 depicts this real-time prediction flow: the user submits a URL through the Gradio interface, which is passed through URL feature extraction, preprocessing and feature selection, the hybrid stacking model (ANN, Bagging KNN, and the Logistic Regression meta-learner), a rule-based verification layer, and finally the Gradio result display.



Real-Time Phishing URL Detection Using Gradio Interface.

B. Browser Extension

A Manifest V3 Chrome extension was developed as a companion client to the Gradio-deployed model, automatically reading the URL of the currently active browser tab and analysing it through a three-tier decision process. First, local rule-based screening extracts the same lexical and structural features as the Gradio interface, reimplemented in JavaScript, and checks the URL against six local rules: trusted-domain whitelist, brand impersonation on free hosting, brand impersonation alone, a high suspicious-keyword count, suspicious TLD combined with suspicious keywords, and IP-address-as-domain. A matching rule returns a verdict immediately with locally defined confidence values, without contacting the trained model. Second, if no rule matches and a Gradio API endpoint has been configured, the URL is sent to that endpoint and the trained model's verdict and probabilities are returned; this is the only path in which the extension's displayed result is produced directly by the trained hybrid model. Third, a local heuristic fallback is used when no endpoint is configured or the remote request fails, computing a locally defined weighted risk score from URL characteristics. The popup interface displays the current tab's URL, a colour-coded verdict banner, confidence bars, a feature-signal grid, a manual URL-checking field, and the API endpoint configuration field.



Phishing Detector Pop-up Interface Showing Classification Result.

Real-Time Browser Extension — Corresponding Confidence Scores (Documented Test Case)

Feature Signal	Observed Value
HTTPS Usage	Present
Free Hosting Detected	Not detected
Brand Impersonation	Not detected
IP-Based Domain	Not detected
Suspicious TLD	Not detected
Suspicious Keyword Count	0
TLD Legitimacy Probability	30%
URL Entropy	4.36
Pred. Confidence — Legitimate	100%
Pred. Confidence — Phishing	0%

The extension automatically detected the active tab's URL and returned a verdict of "Likely Legitimate," with the supporting message "No strong phishing signals found."

C. Deployment Considerations

Because the rule-based screening layer executes before any call to the trained model, a substantial proportion of URLs are classified without model involvement in both interfaces. Additionally, the temporary public link generated by Gradio's sharing feature is time-limited and changes on every relaunch, requiring periodic manual reconfiguration in the extension; if the link becomes stale, the extension silently falls back to local heuristic scoring without on-screen indication of which tier produced a given result.

VII. DISCUSSION

The leakage-conscious preprocessing pipeline, in which the train/test split precedes scaling, class balancing, feature engineering, and feature selection, addresses a common methodological weakness in phishing-detection literature where preprocessing fit on the full dataset can inflate reported test performance. At the same time, the near-perfect accuracy obtained here, consistent with several near-ceiling results reported elsewhere in the literature (ref10?; ref11?; ref13?; ref24?), is interpreted cautiously rather than as unqualified evidence of model superiority, given the presence of highly separable content-based features and an exactly balanced test set that warrants further leakage verification. The dual deployment strategy demonstrates two real-world usage patterns: deliberate URL lookup via Gradio, and passive, automatic screening during browsing via the extension. However, since the extension operates exclusively on URL-derivable features while the offline model benefits from content-based features unavailable at inference time, a meaningful gap likely exists between offline benchmark performance and real-world deployed performance, a distinction this work discusses openly rather than omits. Both interfaces further incorporate a rule-based verification layer that can override or bypass the trained model's prediction for a substantial share of URLs, meaning the hybrid model's contribution to the displayed verdict is smaller than a purely model-driven deployment would suggest.

VIII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

This paper presented a hybrid stacking ensemble for phishing URL detection, combining an ANN and Bagging KNN as base learners with a Logistic Regression meta-learner, trained using a leakage-conscious pipeline in which all splitting-dependent operations were fitted exclusively on the training partition. The stacking ensemble achieved perfect classification (Accuracy, Precision, Recall, F1-Score, and AUC-ROC all 1.0000) on the held-out test set, a result interpreted with explicit caution regarding content-based feature separability and test-set class balance rather than presented as unqualified superiority. The trained model was deployed through a Gradio web interface and a companion Chrome browser extension, together demonstrating both on-demand and passive real-time detection use cases, validated through a documented live test case on the browser extension.

B. Limitations

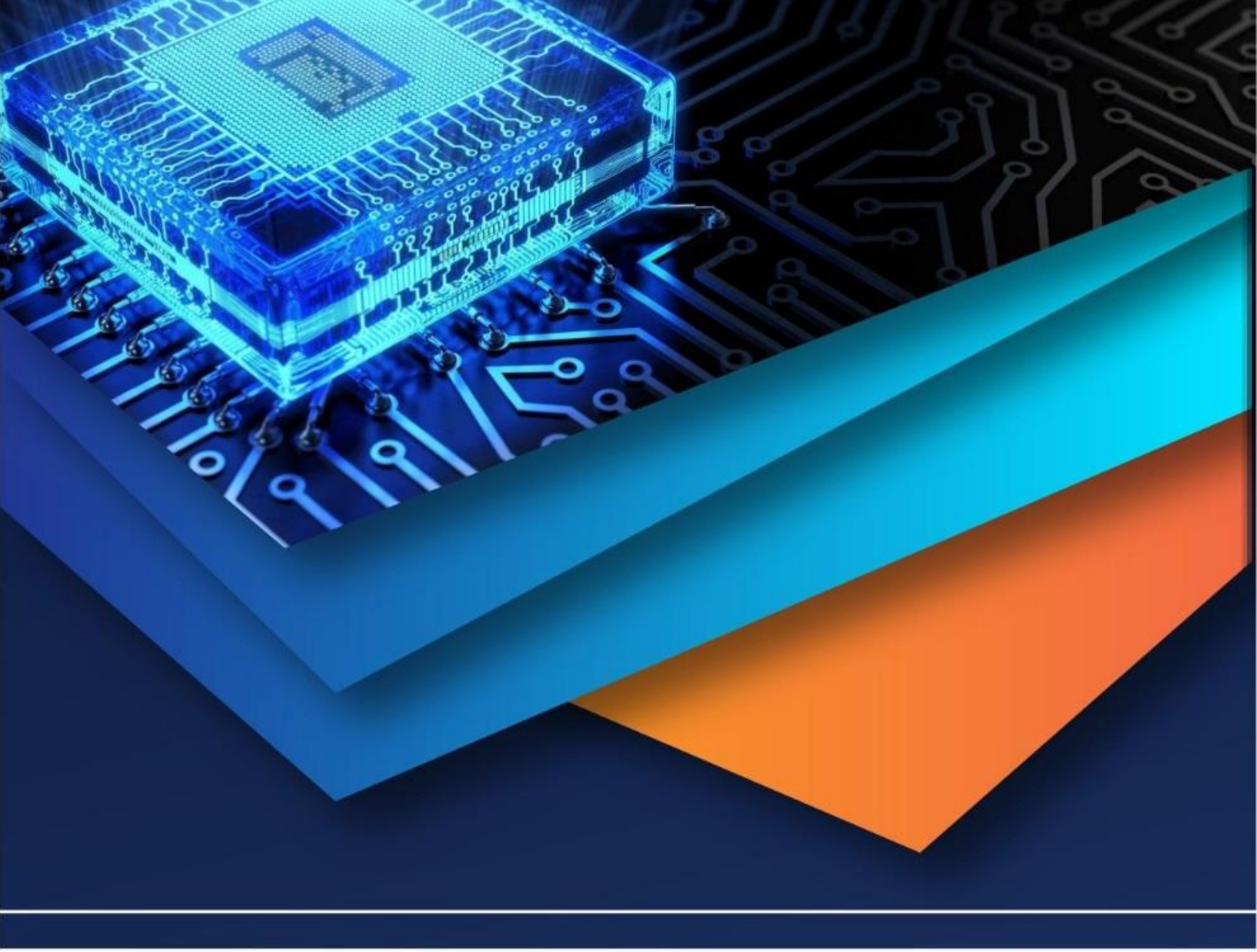
- The offline model's near-perfect accuracy is likely influenced by content-based features (page title, favicon presence, domain-title matching) that cannot be computed from a URL string alone, creating a gap between offline benchmark performance and real-time browser extension performance.
- The PhiUSIIL dataset represents a snapshot of phishing and legitimate URLs at a fixed point in time and may not capture newer obfuscation techniques.
- The rule-based safeguards rely on manually maintained lists requiring periodic updates.
- The system has not been evaluated against adversarially crafted phishing URLs designed specifically to evade the trained model.
- The exactly balanced test set (5,875/5,875) warrants independent re-verification to confirm class balancing was applied strictly to the training partition.

C. Future Scope

- Feature-restricted retraining for deployment parity, using only URL-derivable features so offline metrics reflect achievable browser-extension accuracy.
- Persistent hosted deployment, replacing the temporary Gradio public link with a persistent hosted API endpoint.
- Verdict-source transparency, adding a visible indicator showing whether a verdict originated from the rule layer, the trained model, or the local heuristic fallback.
- Adversarial robustness evaluation against adversarially crafted phishing URLs.
- Independent leakage verification of the SMOTE-based class balancing step, given the exactly balanced test set observed in current results.

REFERENCES

[1] S. Naseeb, S. Ramzan, A. Raza, M. S. A. Hashmi, Y. H. Gu, M. Syafrudin, and N. L. Fitriyani, "Website phishing attack detection using innovative meta-learning based ensemble approach," IEEE Access, vol. 13, pp. 164249–164264, 2025.
[2] P. Jain, R. Sharma, and A. Gupta, "Phishing website detection using ensemble machine learning techniques," International Journal of Computer Applications, vol. 185, no. 12, pp. 10–16, 2025.
[3] D. T. H. Tham, "Meta-ensemble learning model for phishing website detection," Journal of Cybersecurity and Privacy, vol. 5, no. 2, pp. 145–158, 2025.
[4] S. Giri and S. Banerjee, "Greedy stacking ensemble model for phishing website detection," International Journal of Information Security, vol. 23, pp. 311–325, 2024.
[5] P. T. Duy, T. Nguyen, and L. Pham, "Multimodal learning framework for phishing attack detection using GAN-based adversarial training," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 2241–2255, 2024.
[6] M. Dubey, A. Tripathi, A. Srivastava, and S. Singh, "Phishing detection system: An ensemble approach using character-level CNN and feature engineering," arXiv preprint, 2025.
[7] R. Ji, Y. Zhang, and Q. Wang, "Stacked ensemble model with autoencoder for intrusion detection," Computers & Security, vol. 130, 2025.
[8] S. I. Nova, M. Rahman, and T. Ahmed, "Multi-feature extraction with ensemble learning for phishing website detection," International Journal of Advanced Computer Science and Applications, vol. 15, no. 4, pp. 233–241, 2024.
[9] J. Patni, A. Khandelwal, and S. Verma, "Natural language processing based phishing website detection," Procedia Computer Science, vol. 215, pp. 521–528, 2025.
[10] A. Rahmadeyan, H. Al-Qudah, and S. Khan, "Phishing website detection using artificial neural network and AdaBoost," Journal of Information Security and Applications, vol. 73, 2023.
[11] Z. Alamri, A. Alharbi, and M. Alotaibi, "Optimized ensemble stacking model for phishing website detection," Future Internet, vol. 17, no. 1, pp. 1–18, 2025.
[12] A. Shabbir, M. Iqbal, and F. Khan, "Stacking deep learning models for intelligent pattern detection systems," IEEE Access, vol. 12, pp. 11034–11046, 2024.
[13] H. A. Hilal, "Deep learning-based classification for phishing URL detection," International Journal of Cybersecurity Intelligence & Cybercrime, vol. 8, no. 1, pp. 45–56, 2025.
[14] J. Jesmitha, R. Kumar, and P. Reddy, "Machine learning-based phishing detection with real-time email alert system," International Journal of Engineering Research and Technology, vol. 14, no. 3, pp. 210–217, 2025.
[15] M. Patel, S. Shah, and D. Mehta, "Phishing URL detection using machine learning algorithms," Journal of Network and Computer Applications, vol. 215, 2025.
[16] K. L. Chihnavi, R. Ramesh, and A. Kumar, "Ensemble machine learning framework for phishing website detection," International Journal of Advanced Research in Computer Science, vol. 16, no. 2, pp. 98–105, 2025.
[17] S. Kavya and D. Sumathi, "Hybrid artificial intelligence model for phishing website detection using genetic algorithm," Expert Systems with Applications, vol. 234, 2024.
[18] T. Gandhi, R. Sharma, and P. Agarwal, "BLPDS: Deep learning framework for phishing webpage detection," IEEE Access, vol. 12, pp. 22145–22158, 2024.
[19] K. V. Deshpande and J. Singh, "A systematic review of machine learning techniques for phishing detection," ACM Computing Surveys, vol. 57, no. 2, 2025.
[20] H. Saini and N. Kaur, "Ensemble machine learning approach for phishing URL detection," International Journal of Information Security Science, vol. 14, no. 1, pp. 55–64, 2025.
[21] Prabavathi T. and M. M., "A hybrid deep learning and ensemble framework for real-time phishing website detection using optimized feature selection," in Proc. 2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI), 2025.
[22] A. Sharma and A. Rajput, "A hybrid ensemble learning framework for efficient and real-time phishing website detection using optimized URL and domain features," Dandao Xuebao (Journal of Ballistics), 2026.
[23] S., Gokulnath K., Irfan Mohamed, Manikandan M., and R. A., "Cross-browser real-time phishing website detection framework using behavioral analysis and machine learning," Indian Journal of Computer Science and Technology, 2026.
[24] P. S., S. R., and Thirishya M., "HPD: A hybrid ML system for real-time phishing website detection," International Journal of Innovative Science and Research Technology, 2025.
[25] A. Kulaglic and M. A. B. Al-Tarawneh, "Adaptive phishing website detection using incremental machine learning: A dynamic approach to cybersecurity threats," International Journal of Advanced Computer Science and Applications, 2026.
[26] N. Alsuqayh, A. Mirza, and A. Alhogail, "Exploring feature engineering and explainable AI for phishing website detection: A systematic literature review," International Journal of Electrical and Computer Engineering, 2025.
[27] Shammi L. and Emilin Shyni C., "Stacking ensemble classifier for phishing detection: A cyber security model with efficient feature descriptor," International Journal of Intelligent Decision Technologies, 2025.
[28] K. Jishnu and B. Arthi, "Real-time phishing URL detection framework using knowledge distilled ELECTRA," Automatika, 2024.
[29] A. S. Araujo Arévalo, G. A. Felix-Diaz Monzon, and J. P. Mansilla Lopez, "Intelligent system for phishing detection on web pages using random forest," in Proc. LACCEI International Multi-Conference for Engineering, Education and Technology, 2023.
[30] S. Alnemari and M. Alshammari, "Detecting phishing domains using machine learning," Applied Sciences, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY