

# A Framework for Real-Time Phishing Resource Locator Detection Using Structural and Lexical Feature Analysis Based on Machine Learning

Prashant Srivastava<sup>1</sup>, Akhilesh Singh<sup>2</sup>, Amit Virmani<sup>3</sup>, Himanshu Shukla<sup>4</sup>, Ravi Kant Mishra<sup>5</sup>, Ritesh Agarwal<sup>6</sup>, Anand Kumar Mishra<sup>7</sup>

<sup>1-7</sup>School of Engineering and Technology (UIET), CSJM University Kanpur, U.P., India

**Email:** <sup>1</sup>[theprashant1@gmail.com](mailto:theprashant1@gmail.com), <sup>2</sup>[akhileshsingh@csjmu.ac.in](mailto:akhileshsingh@csjmu.ac.in), <sup>3</sup>[amitvirmani@csjmu.ac.in](mailto:amitvirmani@csjmu.ac.in),  
<sup>4</sup>[himanshushukla19@gmail.com](mailto:himanshushukla19@gmail.com), <sup>5</sup>[ravikantmishra2307@gmail.com](mailto:ravikantmishra2307@gmail.com), <sup>6</sup>[riteshagarwal@csjmu.ac.in](mailto:riteshagarwal@csjmu.ac.in),  
<sup>7</sup>[mishra.anand13@gmail.com](mailto:mishra.anand13@gmail.com)

## Abstract

Phishing attacks remains a leading, rapidly evolving threat in cybersecurity domain, where cyber-criminals deploy fraudulent websites that are deceptive in nature acts exactly same as legitimate platforms to transfer sensitive user credentials, financial data and corporate data to an external location. Traditional counter measures including blacklist-based systems and rule-based filters demonstrates limited efficiency against newly created malicious URLs and real-time generated phishing URLs. This paper presents **PhishGuard+**, a random forest-based phishing detection framework that performs real-time suspicious URL classification through complete structural and lexical feature analysis of the URL. The proposed system utilizes a dataset of 10,000 labeled URLs, from which 48 distinct features are extracted that includes URL patterns, domain characteristics, and heuristic indicators. A Random Forest classifier is trained on preprocessed data comprises min-max normalization and the Synthetic Minority Oversampling Technique (SMOTE) to address skewed data distribution. The trained model is deployed with a Flask-based RESTful API, that real-time inference through an interactive web interface. Experimental evaluation demonstrates an overall classification accuracy of 98.4%, with precision and recall exceeding 98% across both phishing and legitimate URL categories. The proposed framework offers a computationally efficient, low latency lightweight detection solution deployable in browser extensions, email security gateways, and corporate cybersecurity infrastructures, providing robust early-warning capabilities without requiring full webpage content analysis.

## Keyword

Phishing detection, malicious URL classification, random forest classifier, lexical feature extraction, SMOTE oversampling, real-time cybersecurity, Flask API

## 1. Introduction

The exponential growth of online services, digital payment systems and cloud-based platforms has

structurally changed the global communication and commerce. However, this growth has simultaneously created an unmonitored entry points for

cyber-criminals, with phishing emerging as one of the most damaging forms of information security. Phishing constitutes a cybercrime technique in which opponent deceive users into visiting fraudulent websites engineered to look exactly as legitimate platforms, thereby conduct the breach of data in the form of authentication credentials, banking information, and personally identifiable data. According to published cybersecurity reports, phishing accounts for over 90% of social engineering attacks and consistently ranks among the primary initial access vectors exploited in enterprise security breaches [1].

Traditional phishing detection mechanisms, that includes URL blacklists, rule-based filters, and signature-based detection systems, have demonstrated significant limitations in the modern threat environment. Cyber-criminals are continuously developing new and different phishing URLs capable of evading static detection systems. Consequently, machine learning (ML) and artificial intelligence (AI) paradigms have gained substantial research attention as feasible alternatives, capable of identifying phishing websites through pattern recognition in URL structures, domain behavioral characteristics, and heuristic web content features.

The primary challenges motivating this research includes:

- Cybercriminals continuously generate new and different phishing URLs that structurally bypass rule-based and signature-based detection systems..
- End-users lack the technical knowledge to reliably distinguish phishing websites from legitimate ones.
- Blacklist-based systems fail against newly active or aged URLs as they are deployed before being blacklisted.
- There exists a shortage of lightweight, dynamic, browser-compatible phishing detection tools suitable for wide area deployment.

To address these challenges, this paper proposes PhishGuard+, a machine learning framework designed to analyze and classify URLs in real time. The principal contributions of this work are as follows:

1. Development of a complete 48-feature extraction that includes lexical, structural, domain-based, and heuristic URL attributes.
2. Training and optimization of a Random Forest classifier achieving 98.4% classification accuracy on a balanced 10,000-URL dataset.
3. Integration of the trained model into a Flask-based RESTful API enabling real-time phishing URL prediction.
4. Construction of an accessible web interface that interacts with user and show the results.
5. Observe performance using accuracy, precision, recall, F1-score, and confusion matrix analysis.

## 2. Literature Review

Early reasearch in the phishing detection domain predominantly focused on heuristic-based URL analysis, examining structural attributes including URL length, the presence of '@' symbols, numeric characters in the domain, and the count of dots in the hostname [1]. While computationally efficient, these early solutions offered limited classification accuracy due to their inability to react across diverse phishing strategies.

Subsequent research introduced feature-rich detection approaches exploiting lexical, host-based, and webpage content-based attributes. Sahingoz et al. [3] demonstrated that deep learning architectures, specifically Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN), could achieve approximately 97% accuracy when trained on 35,000 URLs using character-level and lexical pattern features. However, these approaches incur more computational power, limiting their success in resource-constrained environments.

Aljofey et al. [4] proposed a character-level Convolutional Neural Network for phishing detection, achieving approximately 97.8% accuracy on a dataset of 60,000 URLs using token-based word embeddings. Despite high classification performance, the deep learning model presented interpretability challenges that limit forensic applicability. Rao and Pais [5] evaluated Random Forest and Gradient Boosting classifiers on an 11,000-URL dataset achieving ~96% accuracy through URL structure and domain feature analysis, though without real-time deployment capability.

The PhishStorm framework [2] addressed real-time detection requirements using streaming analytics with ML, processing large-scale URL datasets. This work underscored the importance of deploying phishing detection systems capable of operating at network speed. Table I provides a structured comparison of representative prior studies.

**Table 1.** Comparison of Prior Phishing Detection Studies

| Author / Year                     | Method Used                                      | Features Used                                | Dataset Size            | Accuracy      | Limitations                                             |
|-----------------------------------|--------------------------------------------------|----------------------------------------------|-------------------------|---------------|---------------------------------------------------------|
| Abu-Nimeh et al. (2007)           | ML classifiers (SVM, Decision Tree, Naïve Bayes) | URL and email features                       | 2000+ phishing emails   | ~92%          | Limited feature engineering and small dataset           |
| Marchal et al (2014) – PhishStorm | Streaming analytics with ML                      | URL lexical features and network behavior    | Large real-time dataset | ~95%          | High computational complexity                           |
| Sahingoz et al (2019)             | Deep Learning (LSTM, CNN)                        | URL characters and lexical patterns          | 35,000 URLs             | ~97%          | Requires high training time and computational resources |
| Aljofey et al (2020)              | Deep Neural Networks                             | URL tokens and word embeddings               | 60,000 URLs             | ~97.8%        | Difficult to interpret model decisions                  |
| Rao & Pais (2019)                 | Random Forest and Gradient Boosting              | URL structure and domain features            | 11,000 URLs             | ~96%          | Does not support real-time deployment                   |
| <b>Proposed - PhishGuard+</b>     | <b>Random Forest (Ensemble)</b>                  | <b>48 lexical domain, heuristic features</b> | <b>10,000 URLs</b>      | <b>~98.4%</b> | <b>Lightweight; SMOTE-balanced; Flask-deployed</b>      |

The review reveals that while deep learning methods achieve marginally higher accuracy, they require more computational

resources and lack real-time deployment feasibility. In contrast, ensemble learning methods such as Random Forest offer a more balance between accuracy, interpretability, and efficiency, motivating its selection in the proposed PhishGuard+ framework.

### 3. Proposed Methodology and System Architecture

PhishGuard+ adopts a machine learning-driven workflow architecture organized into six functional stages: dataset acquisition, data preprocessing, feature engineering, model training and optimization, backend API integration, and frontend interface development. This structured workflow ensures that the resulting system is secure, explainable, and suitable for real-time deployment.

#### A. System Architecture Overview

The architecture of PhishGuard+ comprises two primary layers: an offline training pipeline responsible for model construction, extracting features and an online integrated pipeline responsible for real-time URL classification. User-submitted URLs goes to the inference pipeline, which calls the feature extraction module, applies the pre-trained scaler, and ask the Random Forest classifier to produce a classification label with an associated confidence score.

#### B. Dataset Acquisition

A benchmark dataset, Phishing\_Legitimate\_full.csv, that is comprised of 10,000 URLs (5,000 phishing and 5,000 legitimate) was employed for model training and evaluation. This dataset provides balanced class representation and includes 48 pre-computed lexical, structural, domain-based, and behavioral features characterizing each URL.

#### C. Data Preprocessing

To ensure data quality and optimal model performance, the following preprocessing operations were applied:

- Missing values is substituted for complete URL records.
- Min-max normalization of numerical feature values.
- Application of the Synthetic Minority Oversampling Technique (SMOTE) to reduce residual class imbalance.
- Removal of non-predictive values including record identifiers and textual labels.
- Correlation analysis to identify highly correlated feature pairs.

#### D. Feature Engineering

A custom URL feature extractor was implemented to transform raw URL strings into 48-dimensional numerical feature vectors that is aligned with the training dataset. The extracted feature categories consist of:

- Lexical features: URL length, count of dots, hyphens, '@' symbols, numeric characters, and path

depth.

- Domain and hostname features: subdomain count, domain length, hostname structural patterns, and IP address indicators.
- Query and path features: parameter count, path segment depth, and presence of suspicious query strings.
- Heuristic features: detection of brand impersonation keywords, sensitive terms (e.g., 'login', 'secure', 'update'), and suspicious top level domain patterns.
- WHOIS-based features: domain registration age, registrar characteristics, and expiration timeline.

## E. Algorithm Pseudocode

Algorithm 1 presents the pseudocode for the PhishGuard+ URL classification procedure, illustrating the complete inference pipeline from URL input to classification output.

### Algorithm 1: PhishGuard+ Random Forest Phishing Detection

Input: URL string U

Output: Label  $L \in \{\text{Phishing, Legitimate}\}$ , Confidence Score C

```
1. BEGIN
2. Load trained model M, scaler S, feature list F
3. Parse URL U into components: {scheme, domain, path, query, fragment}
4. FOR each feature  $f_i$  in F ( $i = 1$  to 48):
5.     Extract  $f_i$  from U using lexical/domain/heuristic rules
6. END FOR
7. Construct feature vector  $X = [f_1, f_2, \dots, f_{48}]$ 
8. Apply normalization:  $X_{\text{scaled}} = S.\text{transform}(X)$ 
9. FOR each tree  $T_j$  in Random Forest M ( $j = 1$  to N):
10.     $\text{Vote}_j = T_j.\text{predict}(X_{\text{scaled}})$ 
11. END FOR
12.  $L = \text{majority\_vote}(\text{Vote}_1, \dots, \text{Vote}_N)$ 
13.  $C = (\text{count of trees voting for } L) / N \times 100$ 
14. Return  $\{L, C, \text{top\_features}(X, M)\}$ 
15. END
```

## 4. Dataset and Feature Engineering

The dataset used in this research consists of 10,000 labeled URL samples, achieving a balanced 50:50 ratio between phishing and legitimate values. This balanced distribution is critical for preventing classifier bias and ensuring equitable evaluation across both target classes. The dataset consist of samples collected from established phishing darabases and verified legitimate web domains, ensuring representative coverage of real-world URL patterns.

Table II gives a categorized summary of the 48 engineered features utilized in model training, organized by feature type and extraction methodology.

### A. Feature Category Distribution

- Lexical URL features (18 features): Derived from raw URL string analysis including length metrics, character frequency counts, and structural anomaly indicators.
- Domain and host-based features (12 features): Derived from domain name parsing

including subdomain depth, domain entropy, and hostname pattern analysis.

- Query and path features (8 features): Quantifying URL path complexity, parameter multiplicity, and suspicious query string patterns.
- Heuristic and semantic features (6 features): Keyword-based indicators detecting brand impersonation, credential harvesting terms, and social engineering language.
- WHOIS-based domain features (4 features): Domain age, registration duration, registrar reputation indicators, and expiration proximity metrics.

The feature extractor function processes any inconsistent URL string and generates a 48-dimensional feature vector through a sequential parsing workflow. Feature values are subsequently normalized using a pre-fitted MinMaxScaler to ensure consistent input distributions aligned with the training data values.

## 5. Implementation and Tools

The PhishGuard+ system was implemented using a Python-centric technology stack, with dedicated components addressing machine learning model training, backend API development, and frontend user interface construction.

### A. Development Environment

- Python 3.10: Primary language for machine learning, feature extraction, WHOIS lookup, and backend API logic.
- Scikit-learn: Applied for Random Forest classifier training, performance evaluation, and feature importance extraction.
- Imbalanced-learn (SMOTE): Applied for synthetic oversampling of minority class samples to address dataset imbalance.
- Pandas and NumPy: Utilized for data preprocessing, numerical operations, and dataset manipulation.
- Flask: Lightweight WSGI-compatible web framework used to expose the trained ML model through a /predict RESTful API endpoint.
- WHOIS Library: Provides programmatic access to domain registration information for WHOIS-based feature extraction.
- HTML5, CSS3, JavaScript, and Bootstrap 5: Frontend technologies used for user interface design, URL input validation, and dynamic output.
- Visual Studio Code: Primary integrated development environment for system development and debugging.

### B. Module Architecture

PhishGuard+ is organized into six functional modules:

1. Dataset Processing Module: Loads, cleans, applies SMOTE balancing, partitions training and testing splits, and serializes processed feature metadata.
2. Feature Extraction Module: Parses input URLs into structured components and computes the complete 48-dimensional feature vector.
3. Machine Learning Model Module: Trains the Random Forest classifier, evaluates performance metrics, and serializes the model artifact (model.pkl), normalization scaler (scaler.pkl), and feature list (feature\_names.pkl).
4. Backend Prediction API Module: Implements the Flask /predict endpoint, orchestrating feature extraction, scaling, inference, and JSON response generation with confidence scores and feature importance rankings.

5. Frontend UI Module: Provides a responsive web interface for URL submission, result display, confidence visualization, and input validation.
6. Result Visualization and Reporting Module: Generates confusion matrix visualizations, feature importance rankings, WHOIS diagnostic logs, and classification performance reports.

## 6. Experimental Results and Performance Evaluation

Experimental tests was conducted on the held-out test partition comprising 2,000 URL samples (20% of the full dataset) following an 80-20 stratified train-test split. The Random Forest classifier was trained using 100 estimators with default hyperparameters, subsequently refined through grid search optimization. All experiments were executed on a standard workstation environment.

### A. Classification Performance

The trained PhishGuard+ model achieved an overall classification accuracy of 98.4% on the test partition. Table III presents the detailed per-class performance metrics, and Table IV summarizes the confusion matrix decomposition.

**Table 2.** Classification Performance Metrics

| Class                   | Precision    | Recall | F1-Score |
|-------------------------|--------------|--------|----------|
| Legitimate (Class 0)    | 0.98         | 0.99   | 0.98     |
| Phishing (Class 1)      | 0.99         | 0.98   | 0.98     |
| <b>Overall Accuracy</b> | <b>0.984</b> | —      | —        |

**Table 3.** Confusion Matrix Decomposition

| Metric               | Value | Class                 | Interpretation                                     |
|----------------------|-------|-----------------------|----------------------------------------------------|
| True Positives (TP)  | 981   | Phishing              | Correctly identified phishing URLs                 |
| True Negatives (TN)  | 988   | Legitimate            | Correctly classified legitimate URLs               |
| False Positives (FP) | 12    | Legitimate → Phishing | Type I Error – minimize to reduce false alerts     |
| False Negatives (FN) | 19    | Phishing → Legitimate | Type II Error – minimize to prevent missed threats |

### B.Comparative Analysis

Table IV presents a comparative evaluation of PhishGuard+ against alternative machine learning classifiers trained on the same dataset under identical preprocessing environment. The Random Forest ensemble consistently outperforms competing models across all evaluation metrics, attributable to its ensemble learning mechanism that aggregates predictions from multiple decorrelated decision trees, effectively reducing both variance and overfitting approximations.

**Table 4.** Comparative Evaluation of Classification Models

| Model                              | Description                                     | Advantages                                          | Disadvantages                                | Accuracy (Approx.) |
|------------------------------------|-------------------------------------------------|-----------------------------------------------------|----------------------------------------------|--------------------|
| Logistic Regression                | Statistical binary classification model         | Simple, fast and interpretable                      | Cannot capture complex URL patterns          | 88–91%             |
| Naïve Bayes                        | Probabilistic classifier based on Bayes theorem | Fast and efficient on large datasets                | Assumes feature independence                 | 89–92%             |
| Support Vector Machine             | Separates data using optimal hyperplanes        | High accuracy for structured data                   | Computationally expensive for large datasets | 93–96%             |
| Decision Tree                      | Splits data using feature conditions            | Easy to understand and interpret                    | Prone to overfitting                         | 92–95%             |
| Neural Networks                    | Deep learning model for complex patterns        | High predictive capability                          | Requires large datasets and high compute     | 96–97%             |
| <b>Random Forest (PhishGuard+)</b> | <b>Ensemble of multiple decision trees</b>      | <b>High accuracy and robust against overfitting</b> | <b>Slightly slower than simple models</b>    | <b>~98.4%</b>      |

## C.Real-World Validation

Real-world validation was performed by submitting known legitimate URLs (e.g., <https://www.google.com>, <https://www.github.com>) and publicly reported phishing samples to the deployed PhishGuard+ API. The system correctly classified known legitimate domains with confidence scores consistently exceeding 75%, while phishing samples were detected with confidence scores above 90% in the majority of test cases. Occasional reduced confidence scores for legitimate domains were observed in cases where WHOIS server timeouts precluded domain age feature retrieval.

## 6. Discussion

The experimental results confirm that PhishGuard+ achieves state-of-the-art classification performance on the phishing URL detection task, surpassing prior work that consist of traditional ML classifiers and approaching the accuracy of computationally intensive deep learning approaches while maintaining significantly lower inference overhead. The 98.4% accuracy and sub-2% false positive rate demonstrates the framework's practical adaptability for real-world deployment scenarios.

The low false positive rate (FP = 12 out of 1,000 legitimate test samples) is of particular significance, as excessive false alarms degrade user trust and operational utility in security applications. Conversely, the false negative rate (FN = 19 out of 1,000 phishing test samples) represents cases where phishing URLs were misclassified as legitimate, typically involving complex phishing domains with structural characteristics that look exactly as legitimate websites.

A notable limitation identified during evaluation is the dependency on WHOIS-based domain features.

Network timeouts during WHOIS lookup can degrade prediction confidence for newly registered domains, representing a vulnerability in zero-day phishing scenarios. Future iterations of the system should incorporate WHOIS caching mechanisms or fallback feature imputation strategies to maintain robustness under network constraints.

The model's interpretability, facilitated through feature importance rankings exposed with the API response, constitutes a distinct advantage over black-box deep learning alternatives, enabling security analysts to understand the specific URL characteristics contributing to phishing classifications and supporting forensic investigation workflows.

## 7. Applications and Future Work

### A. Deployment Applications

The PhishGuard+ framework is structurally suitable for integration into multiple cybersecurity deployment applications:

- **Browser Extension Integration:** The Flask-based REST API can be called as a browser extension to provide real-time URL detection during web navigation, delivering early security warnings before page content is rendered.
- **Email Security Gateways:** URL extraction and detection can be added into email processing workflows to flag or respond to the messages containing phishing links earlier to user delivery.
- **Enterprise Security Platforms:** The lightweight model can be inserted into Security Information and Event Management (SIEM) systems and network monitoring tools for automatic phishing threat detection.
- **Mobile Application Security:** The model's computational efficiency, low latency enables deployment on mobile endpoint security applications for real-time link verification in messaging and social media applications.

### B. Future Research Directions

Several promising extensions to the current work have been identified:

1. **QR Code Phishing Detection (Quishing):** The growing threat of QR code-embedded phishing URLs (quishing) represents an emerging problem that is not properly addressed by URL text analysis systems. A future enhancement involves integrating a QR code image processing function capable of decoding embedded URLs from the QR images and submitting them to the PhishGuard+ classification pipeline. This extension would extend the system's applicability to physical-world engineering attacks distributed with printed materials, email attachments, and digital advertisements.
2. **Graph Neural Network Integration:** Adding a graph-based representations of URL structural relationships and domain interconnections may improve detection of more complex coordinated phishing efforts.
3. **Multilingual Phishing Detection:** Extending the feature extraction pipeline to handle internationalized domain names (IDN) from any language and then detect

## 8. Conclusion

This paper presented PhishGuard+, a machine learning-based real-time phishing URL detection

framework consisting of a Random Forest classifier trained on 48 engineered structural and lexical URL features. The proposed system provide a solution to the systematic limitations of traditional blacklist-based and rule-based phishing detection techniques through a data-driven classification methodology capable of identifying previously unseen phishing URLs without requiring full webpage content analysis.

Experimental evaluation demonstrated an overall classification accuracy of 98.4%, with precision and recall exceeding 98% across both phishing and legitimate URL categories, outperforming other machine learning classifiers evaluated under identical experimental conditions. The addition of SMOTE-based class balancing and min-max normalization contributed to strong model generalization, while the Flask-based RESTful API deployment enables practical merger into browser extensions, email security gateways, and other cybersecurity platforms.

The high classification performance, computational efficiency, and deployment flexibility of PhishGuard+ establish it as a feasible, sustainable and practical contribution to the cybersecurity ecosystem. Future work will focus on extending the system's detection capabilities to QR code phishing, improving adversarial robustness, and evaluating this framework's performance under large-scale URL handling in real-world conditions.

## References

1. M. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in Proc. Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit, Pittsburgh, USA, 2007.
2. S. Marchal, J. François, R. State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Trans. Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.
3. O. K. Sahingoz, B. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019.
4. A. Aljofey, Q. Jiang, Q. Qu, M. Huang, and J. P. Niyato, "An effective phishing detection model based on character-level convolutional neural network," *Electronics*, vol. 9, no. 9, pp. 1514, 2020.
5. R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing and Applications*, vol. 31, pp. 3851–3873, 2019.
6. Scikit-learn Developers, "Scikit-learn: Machine learning in Python," [Online]. Available: <https://scikit-learn.org>
7. Python Software Foundation, "Python language reference, version 3.10," [Online]. Available: <https://www.python.org>
8. Flask Development Team, "Flask web framework documentation," [Online]. Available: <https://flask.palletsprojects.com>
9. WHOIS Domain Lookup, "WHOIS domain information service," [Online]. Available: <https://www.whois.com>
10. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
11. L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.