International Journal of Scientific Research in Engineering and Management (IJSREM)
Conference - Volume 10 NCET -2026 SJIF Rating: 8.659 ISSN: 2582-3930

An Explainable and Feature-Efficient Machine-Learning Framework for Real-Time Phishing URL Detection

M. Madhumitha[1], P. Mahalakshmi[2], T. Priyadharshini[3], M. Akila[4]
Department of Information Technology, Syed Ammal Arts and Science College, Ramanathapuram, India
[1]madhumithamadasamy36@gmail.com | [2]pmaha2024@gmail.com | [3]pd20050615@gmail.com | [4]akila12102006@gmail.com

Abstract— Phishing URLs remain a practical cybersecurity problem because attackers can imitate legitimate web addresses while attempting to obtain credentials, financial information, or other sensitive data. This paper proposes an explainable and feature-efficient machine-learning framework for real-time phishing URL detection. The framework extracts lexical and structural characteristics from URLs, applies feature-selection methods to reduce computational cost, and combines predictive models with explainability techniques that identify factors contributing to each classification. Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest are selected as baseline models. The protocol evaluates accuracy, precision, recall, F1-score, false-positive rate, inference latency, explanation latency, and resource use. A Python prototype is designed to classify a URL as legitimate or phishing and return an interpretable explanation. The manuscript primarily defines the research design and analysis protocol; the numerical results retained from the source are marked for verification rather than treated as independently validated.

Index Terms— phishing detection, malicious URLs, machine learning, explainable artificial intelligence, feature selection, lexical features, structural features, real-time cybersecurity.

I. INTRODUCTION

Phishing attacks use deceptive websites and URLs to persuade users to disclose sensitive information. URL-based machine learning is attractive because it can operate before a user loads the complete webpage and can avoid the computational and privacy costs associated with retrieving page content [1]. Prior URL-focused work has emphasized lexical analysis, feature selection, and classical machine-learning models as practical components of phishing detection systems [8], [9].

A central design tension is that a detector must identify malicious URLs while remaining fast and understandable. Large feature sets can increase computational cost and introduce redundant or unstable predictors, whereas an overly compact feature set may lose useful information. Recent studies have therefore investigated information gain, mutual information, chi-square, TreeSHAP, SHAP, and LIME for feature selection and interpretation [3], [6]–[8].

This paper proposes a reproducible framework with four objectives: (1) extract inexpensive lexical and structural URL features; (2) compare Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest classifiers; (3) identify a compact feature subset without compromising held-out performance; and (4) produce human-readable explanations while measuring prediction and explanation latency. The contribution is methodological: the paper defines the data pipeline, feature schema, model-comparison protocol, evaluation metrics, explanation procedure, and deployment prototype required to test the framework.

II. RELATED WORK

2.1 URL-Based Phishing Detection

URL-only detection is useful when a decision must be made before page content, scripts, or network behaviour are available [9], [10]. URL-based approaches are attractive for resource-constrained devices because page-content retrieval and external features can be time-consuming and computationally expensive. Lexical methods represent URL strings directly, including character- or n-gram-derived statistical features [7], [9]. Common variables include total length, token length, number of subdomains, IP-address usage, protocol, suspicious terms, special characters, encoding patterns, and domain organization [9], [10].

2.2 Feature Selection and Efficiency

Feature selection can reduce inference cost, simplify explanations, and limit reliance on redundant variables [3], [7], [8]. Prior studies have compared information

gain, information-gain ratio, chi-square, TreeSHAP, and related approaches. The proposed study compares filter-based and model-informed selection under a common validation protocol. Selection is performed inside training folds rather than before cross-validation to prevent validation information from influencing the selected feature set.

2.3 Explainable Phishing Detection

SHAP estimates feature contributions relative to a model prediction baseline, while LIME constructs a local approximation around an individual prediction [2], [4]. In phishing detection, these methods can expose URL properties associated with a phishing prediction. E2Phish combines ensemble models with mutual-information selection and uses SHAP, LIME, and partial-dependence plots for interpretation [6]. Post-hoc explanations are treated as descriptions of model behaviour, not proof of causality. The proposed protocol therefore assesses explanation stability, agreement with selected features, and usefulness for analyst review.

III. PROPOSED FRAMEWORK

3.1 System Architecture

The framework contains five stages: (1) Input—receive a URL from a browser extension, API, gateway, or Python command-line interface; (2) Normalization—parse the URL while preserving scheme, host, path, query, fragment, encoding, and delimiters; (3) Feature extraction—calculate lexical and structural features without requesting the target webpage; (4) Classification—apply the trained model and return the predicted class and confidence score; and (5) Explanation—identify features contributing most to the prediction and return a concise analyst-readable explanation.

The production path should avoid features requiring page retrieval, DNS resolution, or third-party reputation services unless those dependencies are explicitly included in a separate evaluation. This keeps the proposed system focused on fast URL-only detection.



Fig. 1. Proposed system architecture for explainable and feature-efficient phishing URL detection.

3.2 Feature Design

The initial feature schema contains six groups: length features (URL, hostname, path, query, and fragment lengths); token and delimiter features (dots, hyphens, slashes, question marks, equal signs, ampersands, percent signs, at-signs, and related symbols); host structure (subdomains, hostname tokens, IP addresses, nonstandard ports, punycode and internationalized-domain indicators); scheme and encoding features; lexical indicators such as suspicious, login, verification, and urgency terms; and path/query structure including path segments, parameter counts, query-to-URL length ratio, and suffix indicators. The suspicious-term vocabulary will be documented with the dataset version, and sensitivity analyses should compare general and masked dictionaries.

3.3 Data Preparation

The study uses a public labelled dataset containing legitimate and phishing URLs, supplemented only when inclusion criteria and labelling can be documented. Duplicate URLs and exact near-duplicates are removed before splitting. Where domain or campaign metadata are available, grouped splits are added as a robustness analysis. The primary split is stratified into training, validation, and held-out test partitions. Scaling, vocabulary construction, feature-selection thresholds, and class-balance operations are fitted using training data only; the final test set remains untouched until the model and feature subset are frozen.

3.4 Candidate Models

Model	Main purpose
Logistic Regression	Linear baseline
SVM	High-dimensional classification
Decision Tree	Interpretable decision rules
Random Forest	Robust ensemble classification

3.5 Feature-Selection Protocol

1) Full set: all features passing data-quality checks.

2) Filter-selected set: training-fold-only mutual information, information gain, or chi-square ranking.

3) Model-informed set: model importance or SHAP values computed on training data.

Candidate subset sizes are selected before testing—for example, 10%, 25%, 50%, and 75% of the full feature set together with the complete set. Feature-selection

stability is assessed by recording how often each feature is selected across repeated training folds. A feature that is highly ranked in one split but rarely selected elsewhere is treated as unstable rather than a definitive indicator.

3.6 Explainability Protocol

For each final model, the system generates a global feature-importance ranking, a local explanation for each test URL, the direction of each contribution where supported, the predicted class and confidence, and a concise textual summary. SHAP is used for global and local attribution where computationally appropriate; LIME may be used as a secondary local method [2], [4], [6]. Agreement is assessed through top-k feature overlap and representative true-positive, false-positive, true-negative, and false-negative cases. The interface avoids causal language and instead states that a feature contributed positively or negatively to the model prediction.

IV. EXPERIMENTAL DESIGN

4.1 Research Questions

RQ1: Which candidate classifier provides the best balance of phishing recall, precision, F1-score, false-positive rate, and latency? RQ2: Can feature selection reduce the number of URL features and computational cost without materially degrading held-out performance? RQ3: Which lexical and structural feature groups contribute most strongly to model predictions? RQ4: Are explanations stable across data splits, models, and explanation methods? RQ5: Can the final pipeline provide a prediction and interpretable output within a latency target suitable for real-time screening?

4.2 Evaluation Metrics

1) Accuracy: proportion of all URLs classified correctly.

2) Precision: proportion of URLs predicted as phishing that are actually phishing.

3) Recall: proportion of phishing URLs detected.

4) F1-score: harmonic mean of precision and recall.

5) False-positive rate: legitimate URLs incorrectly classified as phishing.

6) Confusion matrix: counts of true positives, false positives, true negatives, and false negatives.

7) ROC-AUC and PR-AUC, with PR-AUC emphasised when class imbalance is substantial.

8) Prediction latency: time from normalized URL input to returned class and score.

9) Explanation latency: additional time required to produce the explanation.

10) Resource use: peak memory and, where feasible, CPU time.

Latency should be measured over repeated batches after warm-up, with hardware, software versions, batch size, and timing boundaries documented. Median and upper-percentile latency should be reported rather than a single best-case measurement.

4.3 Baselines and Ablations

1) Full features versus selected features.

2) Lexical-only versus structural-only versus combined features.

3) Each classifier with and without class weighting or resampling.

4) SHAP/LIME explanations versus intrinsic Decision Tree explanations.

5) Random split versus grouped-by-domain or campaign-aware split when grouping information is available.

6) Explanation-enabled versus prediction-only inference for latency comparison.

These ablations are necessary because a high score on a random URL split may reflect duplicated campaigns or domain-specific artifacts rather than generalization to future phishing URLs.

V. PYTHON PROTOTYPE DESIGN

The prototype is designed as a modular Python package:

```
phishing_detector/
├── data.py        # loading, validation, deduplication, splitting
├── features.py    # URL parsing and lexical/structural features
├── training.py    # cross-validation, feature selection, model fitting
├── explain.py     # SHAP/LIME or intrinsic explanations
├── predict.py     # single-URL and batch inference
└── evaluate.py    # metrics, latency, plots, report generation
```

A prediction response should contain the input URL, predicted label, model score, top contributing features and directions, prediction latency, and explanation latency. The deployed interface should log model version, feature-schema version, timestamp, and outcome without unnecessarily storing sensitive URLs. If URLs must be retained for debugging, access control and an explicit retention policy should be applied.

VI. THREATS TO VALIDITY AND ETHICAL CONSIDERATIONS

A public dataset may not represent current phishing campaigns, regional domain patterns, mobile URLs, shortened URLs, or multilingual attacks. Dataset age and collection procedure should therefore be reported, and temporal or campaign-aware evaluation should be used when metadata permit. A fixed random split may overestimate generalization when related URLs occur in both training and test data [1], [5], [9].

Labels may contain errors or reflect inconsistent definitions of phishing. The study should record the source and labelling rule for each class and avoid describing dataset-derived labels as unquestionable ground truth. Performance should be reported separately for different URL families or collection sources when annotations are available.

Explainability also has limitations: feature attribution describes model behaviour for a given input and does not establish that a feature is causal. Correlated features may also substitute for one another. Stability checks, representative error analysis, and disclosure of the explanation method are therefore part of the protocol. Operational deployment should treat the prototype as a screening aid rather than an autonomous authority for blocking users and should define threshold governance, analyst override, model monitoring, and update procedures.

VII. EXPECTED CONTRIBUTIONS

The completed study is expected to contribute a reproducible URL-only phishing-detection pipeline, a controlled comparison of four classical classifiers, an evaluation of feature-selection strategies, and an explanation interface connecting each prediction to model-relevant URL properties. These contributions are supported only if executed experiments demonstrate that the selected feature set preserves performance and improves efficiency under held-out evaluation.

The paper does not assume that explainability improves classification accuracy. Predictive performance, computational efficiency, and interpretability are treated as distinct outcomes because feature selection may improve one while worsening another.

VIII. RESULTS

The source document includes the following model-performance values. Because the manuscript elsewhere states that the study is a research design and analysis plan and does not report executed experimental results, these values are retained as source-reported values but require author verification against the actual dataset, code, and evaluation run before publication.

Model	Accuracy	Recall	F1-Score
Logistic Regression	97.89%	97.83%	98.90%
SVM	98.55%	98.50%	99.25%
Decision Tree	98.49%	99.21%	99.22%
Random Forest	98.70%	99.28%	99.33%

Table 2. Source-reported performance values; experimental verification required.

IX. CONCLUSION

This paper proposes an explainable and feature-efficient framework for real-time phishing URL detection. The design combines lexical and structural URL features, four classical machine-learning classifiers, training-fold-only feature selection, SHAP/LIME or intrinsic explanations, and explicit measurement of discrimination, false positives, resource use, and latency. The protocol is designed to reduce data leakage and expose whether compact models generalize beyond a random split.

The numerical values retained in Section VIII should be treated as provisional until the dataset, implementation, and evaluation pipeline are executed and independently checked. Final deployment claims should therefore be based on held-out testing, robustness analysis, latency measurements, and reproducibility evidence.

DECLARATIONS

Data availability: To be completed after the dataset source, license, version, and preprocessing records are finalized. Code availability: To be completed after the prototype repository and reproducibility materials are prepared. Conflicts of interest: To be completed by the authors. Funding: To be completed by the authors.

REFERENCES

[1] S. S. Shafin, “An Explainable Feature Selection Framework for Web Phishing Detection with Machine Learning,” 2024.

[2] N. Pundlik, T. Saxena, and J. Mungi, “A Scalable Explainable AI Framework for Phishing Attack

Detection with Interpretable Feature Insights with Machine Learning Based Ensemble Models," 2025.

[3] D. R. Patil et al., "Enhanced Phishing URLs Detection using Feature Selection and Machine Learning Approaches," 2024.

[4] R. A. D. Talasari and E. A. Kusnanti, "Phishing URL Detection Using URL Characteristics: A Multi-Algorithm Machine Learning and Explainable AI (XAI) Approach," 2026.

[5] H. K. M. Lai et al., "Explainable Machine Learning Framework for Phishing URL Detection," 2026.

[6] R. Alam et al., "E2Phish: Explainable Ensemble Machine Learning Model for Enhanced Phishing URL Detection," 2024.

[7] M. Alajlouni et al., "Optimizing Phishing URL Detection: A Comparative Analysis of Feature Counts and Training Times," 2024.

[8] L. M. Rani et al., "Feature Selection to Enhance Phishing Website Detection Based On URL Using Machine Learning Techniques," 2023.

[9] M. AbuTaha et al., "URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis," 2021.

[10] B. Banik et al., "Lexical Feature Based Feature Selection and Phishing URL Classification Using Machine Learning Techniques," 2020.

[11] S. Ali et al., "Entropy-Based Feature Selection Classification Approach for Detecting Phishing Websites," 2019.