



ADAPTIVE PHISHING URL DETECTION USING FEATURE STABILITY AND UNCERTAINTY BASED DECISION MAKING INFORMATION TECHNOLOGY

Mr. Suryawanshi A. M.¹, Tule Payal²

¹ Faculty Guide & Supervisor | ² Dattakala Group Of Institutions Swami - Chincholi tal. Daund dist. Pune, India

Corresponding author: payaltule4@gmail.com

ARTICLE ID IJREK-2026-00157	ARTICLE TYPE Research Article	RESEARCH AREA Computer Science	PUBLICATION Volume 1, Issue Issue 1 (2026)
RECEIVED 23 October 2026	REVISED 28 October 2026	ACCEPTED 26 October 2026	PUBLISHED 29 October 2026

Abstract

Phishing URLs are designed to mislead users into visiting deceptive websites and disclosing sensitive information. Blacklist-based systems can identify previously reported addresses, but newly created or modified URLs may not yet have a reputation record. Machine-learning classifiers can identify patterns in URL strings; however, their performance may depend on dataset composition, feature selection, and changes in attacker behaviour. This paper proposes an adaptive phishing URL detection pipeline that combines lexical and structural feature extraction, feature-stability analysis, supervised classification, and uncertainty-based decision making. Feature stability is assessed across repeated or time-separated training samples to identify features that remain consistently informative. The classifier produces a Phishing or Legitimate label only when its confidence satisfies a validation-selected threshold; otherwise, the system returns Uncertain for further review. The paper defines an evaluation protocol using precision, recall, F1-score, ROC-AUC, calibration, coverage, and covered error rate, with domain- and time-aware splits where feasible. This work presents a proposed methodology and research design. The model has not yet been implemented or experimentally evaluated, and no quantitative performance improvement is claimed.

Keywords: Research; Technology

1. INTRODUCTION

Online banking, e-commerce, email, social networking, and cloud services depend on users reaching the intended website. Phishing attacks exploit this trust by presenting URLs or webpages that imitate legitimate services and encourage users to disclose credentials or other sensitive information. A URL can be changed quickly, and newly registered or modified addresses may not immediately appear in reputation feeds or blocklists.

Machine-learning methods can classify URLs using lexical and structural properties such as length, token patterns, subdomain count, suspicious symbols, and the use of IP addresses. Nevertheless, a model that performs well on one dataset may degrade when evaluated on new domains or later collection periods. Dataset leakage, duplicated URLs, class imbalance, and unstable feature selection can make reported performance difficult to generalize.

2. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement Phishing URLs are designed to imitate trusted websites or conceal the destination of a link. A user may encounter such links in email, messaging applications, social media, advertisements, or search results. URL-based detection is attractive because the URL can be inspected before a page is fully loaded. However, attackers can change URL spelling, subdomain structure, path names, encoding, and hosting infrastructure quickly. A detector that relies on a small set of fixed indicators may therefore become less reliable when the data distribution changes.

B. Research Objectives The objectives of this work are: (1) to describe a reproducible URL-based phishing detection pipeline; (2) to examine lexical, structural, host-related, and security-related URL features; (3) to investigate feature stability across data splits or time periods; (4) to compare supervised classification methods using appropriate evaluation measures; (5) to introduce an uncertainty-aware decision policy that can abstain when confidence is insufficient; and (6) to define an

evaluation plan that reports both security performance and the cost of uncertain decisions.

C. Research Questions The proposed study is guided by four questions. RQ1: Which groups of URL features provide useful discrimination between phishing and legitimate URLs? RQ2: How consistently are selected features ranked across repeated splits or time-based partitions? RQ3: How does an uncertainty-based abstention option affect precision, recall, coverage, and false-positive behaviour? RQ4: What limitations arise when a detector is evaluated on a dataset that may not represent future phishing campaigns? These questions are intended to guide experiments rather than imply results that have not yet been measured.

3. BACKGROUND AND RELATED WORK

A. URL-Based Machine Learning Prior work has examined machine-learning approaches that learn patterns from URL strings and associated website characteristics. Sahingoz et al. studied phishing detection from URLs using machine-learning methods [1]. Jain and Gupta investigated client-side phishing website detection using a machine-learning approach [2]. Other research has combined URL and HTML features to represent both the address and the content of a webpage [3]. These studies motivate automated feature extraction, model comparison, and attention to the environment in which detection is performed.

B. Deep Learning and Broader Surveys Deep-learning methods have also been explored for malicious URL and phishing detection [4]. Such methods may learn complex patterns, but their usefulness depends on the quality, scale, and representativeness of the training data. Benchmarking studies and surveys emphasize that reported results can be difficult to compare when datasets, split strategies, feature sets, and metrics differ [5], [6]. A high score on one dataset should not be interpreted as a guarantee of performance against newly emerging URLs.

C. Research Gap Addressed Recent external literature reinforces the need for evaluation beyond random in-dataset splits. Rashid et al. (2024) report that URL phishing detectors can lose performance across datasets and investigate unsupervised

domain adaptation. A 2026 study by Ahamed et al. presents an integrated protocol for testing adversarial robustness, domain generalization, and explanation stability. These works support the importance of cross-dataset and stress testing, but they do not establish that the proposed feature-stability and abstention pipeline in this paper has already been implemented or validated.

This paper focuses on two design considerations that should be evaluated together. First, feature stability asks whether a feature remains useful across repeated samples or across time, instead of being selected only because it performs well on one split. Second, uncertainty-aware decision-making allows the detector to label a case as uncertain rather than forcing every input into a phishing or legitimate class. These are proposed design choices. Their benefits must be demonstrated experimentally and may vary by dataset, model, and threshold.

4. PROPOSED SYSTEM ARCHITECTURE

A. System Overview The proposed system accepts a URL as input and returns a class decision together with a confidence or uncertainty indicator. The pipeline consists of input validation, URL normalization, feature extraction, feature stability analysis during training, model training, probability calibration where appropriate, threshold-based decision-making, and result logging. Training and inference are separated: stability analysis and model selection are performed using training data only, while a held-out test set is reserved for final evaluation.

B. Input Validation and Normalization The input stage checks that a value is present and can be parsed as a URL. The implementation should define how schemes, internationalized domain names, percent-encoding, default ports, fragments, and trailing separators are handled. Normalization must be applied consistently during training and inference. The raw URL should be preserved for audit and reproducibility, subject to privacy and retention rules. Invalid inputs should receive a documented error response rather than silently being converted into a normal URL.

C. Feature Extraction The feature extractor converts the URL into a numeric representation. Lexical features can include total length, host-name length, path length, number of dots, number of hyphens, number of digits, number of special characters, and counts of subdirectories or query parameters. Structural features can include the number of subdomains, whether an IP address is used as the host, the presence of an explicit port, and the presence of an at-sign or unusual delimiter. Keyword features can flag terms often seen in deceptive paths, such as login, verify, update, or account; these terms are indicators rather than proof of maliciousness. If DNS, certificate, or reputation features are included, their availability, collection time, and failure behaviour must be documented.

D. Feature Stability Analysis Feature stability is evaluated on training data by repeating feature ranking or selection across multiple training resamples. For each resample, the selected features and their ranks are recorded. Stability can be summarized using selection frequency, average rank, rank dispersion, or an appropriate set-similarity measure between selected feature sets. Features that appear consistently may be preferred over features that rank highly in only one split. Stability analysis must not use the held-out test labels, because that would leak test information into model development. The selected feature list and analysis settings should be saved so the experiment can be reproduced.

E. Model Training Candidate classifiers may include Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, and gradient-boosted trees, depending on the available software environment and dataset size. A simple baseline, such as a majority-class classifier or a basic lexical model, helps establish whether more complex methods provide meaningful gains. Hyperparameters should be selected using training-only cross-validation. If classes are imbalanced, class weighting or carefully applied resampling may be considered; any resampling must occur within each training fold rather than before splitting the data.

F. Uncertainty-Based Decision Policy Many classifiers produce a class probability or score, but these values are not automatically calibrated probabilities. Where probability estimates are used, calibration should be evaluated on validation data. A simple policy can assign phishing when the estimated phishing probability exceeds an upper threshold, legitimate when it falls below a lower threshold, and uncertain for values between the thresholds. Thresholds should be selected using validation data and a documented cost rationale. The uncertain outcome is an abstention, not a

third ground-truth class. Evaluation should report how many inputs receive an automated decision and how many require review.

G. Inference and Output At inference time, the system validates and normalizes the URL, computes the required features, applies the trained transformation and classifier, and then evaluates the uncertainty policy. The output should include the predicted category, uncertainty status, and a concise explanation of relevant indicators where feasible. Explanations should not disclose sensitive training records or present heuristic indicators as definitive proof. The system should also handle missing host metadata or unavailable external services without crashing, and should record failures in a way that supports later analysis.

5. EXPERIMENTAL DESIGN AND EVALUATION

A. Dataset and Labeling A reproducible experiment requires a documented dataset containing URLs and labels, together with the source, collection period, label definitions, and any filtering applied. The paper should state the number of phishing and legitimate examples after cleaning, how duplicates were handled, and whether labels were independently verified. If multiple URLs originate from the same domain or campaign, random row-level splitting may place highly related examples in both training and test sets. Where feasible, domain-aware or time-based splitting should be considered to better estimate performance on unseen domains or later observations.

B. Data Preparation and Leakage Prevention All preprocessing decisions should be learned from training data only. Duplicate removal, encoding, scaling, feature selection, resampling, and hyperparameter tuning must be placed inside the training pipeline as appropriate. The held-out test set must not influence feature selection, threshold choice, or model selection. If the dataset contains timestamps, the paper should explain whether the test set represents a later period. Any excluded examples and the reasons for exclusion should be reported.

C. Evaluation Metrics Accuracy measures the proportion of all predictions that are correct, but it can be misleading when the classes are imbalanced. Precision measures the proportion of predicted phishing URLs that are truly phishing, while recall measures the proportion of phishing URLs detected. The F1-score summarizes precision and recall using their harmonic mean. Specificity, false-positive rate, confusion matrix, and ROC-AUC can provide additional context. If the model outputs calibrated scores, a calibration measure may also be useful. For the uncertainty policy, report coverage (the proportion of inputs receiving a non-abstaining decision), selective risk among covered cases, and the number of uncertain cases.

D. Comparison Protocol Each candidate model should be evaluated using the same split strategy and feature availability. Report the configuration, random seed where applicable, software versions, and the procedure used to select hyperparameters. Repeated runs or cross-validation can describe variation, while a final held-out test set should be used only after the pipeline is fixed. If runtime is relevant, measure feature extraction time and prediction latency under a stated environment. Comparisons should avoid claiming that one model is universally best based on a single dataset.

E. Results Reporting Template The current manuscript does not contain completed experimental measurements. After implementation, the following table should be populated with measured values from the same held-out test set. Do not fill the cells with estimated or illustrative values. The uncertainty-enabled system should additionally report coverage and selective risk at the chosen thresholds.

6. IMPLEMENTATION PLAN

A. Software Workflow A practical implementation can be developed in Python. Data inspection and cleaning can use pandas; URL parsing can use the standard URL parsing utilities; feature engineering can use pandas and regular expressions; and model pipelines can be implemented with scikit-learn or another clearly documented library. A single preprocessing function should be shared by training and inference to reduce discrepancies. The trained model, feature schema, selected features, calibration object, and decision thresholds should be saved together with version information.

B. Reproducibility and Testing Unit tests should cover malformed URLs, URLs with IP hosts, unusually long paths, percent-encoded strings, missing schemes,

empty values, and internationalized domain names. Pipeline tests should verify that feature names and ordering match the training schema. A reproducibility record should include dataset version or access date, code version, dependency versions, split indices or seed, model parameters, and threshold-selection method. If external reputation or DNS services are used, the implementation should record the query date and define how unavailable responses are handled.

C. Deployment Considerations A browser extension or gateway could use the model to warn users about suspicious links, but deployment introduces additional constraints. Network requests to external services can add latency and disclose URLs to third parties. Local processing may reduce external disclosure but can limit access to real-time domain information. The user interface should communicate uncertainty clearly and avoid implying that a legitimate classification guarantees safety. High-risk environments may require a human review route, monitoring for distribution shift, and a process for updating the model as new examples become available.

7. THREATS TO VALIDITY

A. Dataset Validity A dataset may overrepresent particular domains, languages, attack campaigns, or collection sources. Labels can be outdated or incorrect, and legitimate URLs may later host malicious content or vice versa. These issues limit how far results can be generalized. Dataset provenance and collection period should therefore be reported, and evaluation on temporally separated or independently collected data is desirable.

B. Experimental Validity Performance estimates can be inflated by duplicate URLs, domain overlap, target leakage, or tuning on the test set. Small test sets can also produce unstable estimates. A clear split policy, training-only model selection, duplicate analysis, and reporting of class counts help reduce these risks. If repeated experiments are conducted, the manuscript should report variation rather than only the best run.

C. Operational Validity URL-only detection cannot inspect every signal available to a browser or security gateway. Some attacks use reputable compromised domains, shorteners, redirects, or pages whose malicious behaviour is not apparent from the URL string. Conversely, legitimate URLs can contain long paths, many parameters, or security-related words. The proposed method should therefore be viewed as one component of layered protection, not a replacement for browser safeguards, reputation services, or user education.

8. ETHICAL AND PRIVACY CONSIDERATIONS

A. Data Handling URLs can contain personal identifiers, session tokens, search terms, or internal resource names. Dataset collection and logging should minimize the retention of sensitive query strings and credentials. Public availability of a URL does not automatically remove privacy obligations. Access controls, retention limits, and documented handling procedures should be applied where data could identify individuals or organizations.

B. Risk of False Alarms and Missed Detections False positives can interrupt legitimate work, reduce trust in warnings, or cause users to ignore future alerts. False negatives may allow a phishing link to pass undetected. Threshold selection should reflect the intended deployment and should be evaluated with both error types. An uncertainty option can support cautious handling of borderline cases, but it also creates review workload and must be measured rather than assumed to improve safety.

9. DISCUSSION

A. Interpretation of the Proposed Design The proposed pipeline combines model comparison with an explicit examination of feature stability and decision uncertainty. These elements address different problems: stability analysis asks whether the learned signal is repeatable, while uncertainty handling asks whether the model has enough evidence to make an automatic decision. Neither mechanism guarantees robustness. Stable features can still encode dataset bias, and an uncertainty score can be misleading if model probabilities are poorly calibrated or the input distribution changes.

B. Practical Trade-offs More extensive feature sets can improve discrimination but may require external lookups, increase latency, or introduce privacy concerns.

Simpler lexical features are easier to compute locally but may be less informative for URLs hosted on compromised trusted domains. Abstention may reduce errors among automatically classified cases while lowering coverage. The final system should therefore report a performance-versus-coverage profile rather than presenting a single threshold as optimal for every use case.

10. LIMITATIONS AND FUTURE WORK

A. Limitations This paper specifies a proposed methodology and evaluation plan. It does not claim a completed implementation, a measured accuracy value, or a demonstrated reduction in phishing incidents. Effectiveness remains dependent on the dataset, label quality, split strategy, model calibration, and deployment conditions. Some related-work references and publication details should be verified against publisher records before submission.

B. Future Work Future work should implement the pipeline and evaluate it on a documented public dataset and, where possible, a time-separated test set. Additional experiments could compare lexical-only features with metadata-enriched features, quantify feature stability across time, and plot coverage against selective risk for several uncertainty thresholds. Further work could examine domain-level grouping, concept drift, model update policies, explainability, and integration into a browser warning workflow. Results should include error analysis and examples of both false positives and false negatives, with potentially harmful live URLs handled responsibly.

11. CONCLUSION

This paper proposes an uncertainty-aware framework for phishing URL detection that combines URL feature extraction, feature stability analysis, supervised model comparison, and a three-outcome decision policy. The framework is intended to make model development more reproducible and to expose cases where an automatic classification may be unreliable. Its evaluation should measure conventional classification performance alongside coverage, selective risk, latency, and the impact of false alarms. Since experiments have not yet been completed for this manuscript, no empirical improvement is claimed. A future implementation and carefully controlled evaluation are required to determine whether the proposed design offers practical benefits.

Research Status: This manuscript presents a proposed methodology and reproducible evaluation plan. No implementation or experiments have been completed for this draft; therefore, no measured performance results are claimed.

REFERENCES

- [1] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019, doi: 10.1016/j.eswa.2018.09.029.
- [2] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning approach," *Telecommunication Systems*, vol. 68, pp. 687–700, 2018, doi: 10.1007/s11235-017-0414-0.
- [3] "A stacking model using URL and HTML features for phishing webpage detection," *Future Generation Computer Systems*, vol. 94, pp. 27–39, 2019, doi: 10.1016/j.future.2018.11.004.
- [4] A. Al-Alyan and S. Al-Ahmadi, "Robust URL phishing detection based on deep learning," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 7, pp. 2752–2768, 2020, doi: 10.3837/tiis.2020.07.001.
- [5] "An in-depth benchmarking and evaluation of phishing detection research for security needs," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2969780.
- [6] "Phishing or not phishing? A survey on the detection of phishing websites," *IEEE Access*, vol. 11, pp. 18499–18519, 2023, doi: 10.1109/ACCESS.2023.3247135.
- [7] S. K. Hasane Ahammad et al., "Phishing URL detection using machine learning methods," *Advances in Engineering Software*, vol. 173, article 103288, 2022, doi: 10.1016/j.advengsoft.2022.103288.
- [8] G. H. Lokesh et al., "Phishing website detection based on effective machine learning approach," *Journal of Cyber Security Technology*, 2021, doi: 10.1080/23742917.2020.1813396.
- [9] Y. Tian, Y. Yu, J. Sun, and Y. Wang, "From past to present: A survey of malicious URL detection techniques, datasets and code repositories," *Computer Science Review*, vol. 58, article 100810, 2025, doi: 10.1016/j.cosrev.2025.100810.
- [10] F. Rashid, B. Doyle, S. C. Han, and S. Seneviratne, "Phishing URL detection generalisation using Unsupervised Domain Adaptation," *Computer Networks*, vol. 245, article 110398, 2024, doi: 10.1016/j.comnet.2024.110398.
- [11] T. Ahamed et al., "An integrated evaluation protocol for adversarial robustness, generalization, and explanation stability in URL-based phishing detection," *Frontiers in Computer Science*, vol. 8, article 1834407, 2026, doi: 10.3389/fcomp.2026.1834407.
- [12] A. Chaudhuri and B. Mohankumar, "Adversarial-resilient lightweight phishing URL detection: Evaluating lexical & metadata features under evasion techniques," *Scientific Reports*, vol. 16, article 26668, 2026, doi: 10.1038/s41598-026-60046-3.

Publication archival note: Official scholarly contribution published under CC BY 4.0 International License. Permanent record on <https://www.ijrek.org> | Article ID: IJREK-2026-00157