Journal of Advanced Intelligent Computing and Informatics
(JAICI)

Volume 1 Issue 1
Article Number: JAICI20260005

Machine Learning-Based Classification of Phishing Websites Using URL and Domain Features

Balakrishna S. Maddodi[1] and Nithesh Naik* [2]

[1]School of Civil & Chemical Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India
[2]Research and Innovation, European City University, Paris, France

Abstract

Phishing websites imitate trusted services to obtain credentials, financial information, or other sensitive data. Blacklists cannot reliably recognize newly created URLs, whereas content-based systems may require page retrieval, rendering, or third-party queries that add latency, exposure, and reproducibility concerns. This paper proposes a lightweight conceptual framework that classifies a URL before visiting its webpage by using 16 interpretable lexical and domain-related features. The framework covers data cleaning, duplicate removal, feature extraction, leakage-controlled splitting, model fitting, evaluation, and feature interpretation. A reproducible proof-of-concept simulation uses the open PhiUSIIL v2 URL corpus, with only its raw URL and label fields. After removing 425 exact duplicate URLs, 235,370 records were split 80:20 with stratification and a fixed seed. Logistic Regression, a depth-limited Decision Tree, and a depth-limited Random Forest were evaluated without exhaustive tuning. On the 47,074-record test set, Random Forest obtained accuracy 0.995943, precision 0.999098, recall 0.991395, F1-score 0.995231, specificity 0.999333, and ROC-AUC 0.998037. These values establish feasibility on this particular random split, not universal detection capability. The contribution is an auditable baseline linking simple Python feature extraction to conventional classifiers and deployment-oriented metrics. Major limitations include possible domain-family overlap, dataset-specific artifacts, temporal drift, and omission of webpage, certificate, redirection, and behavioral evidence.

Keywords: Phishing Detection; Malicious URLs; Machine Learning; Cybersecurity; URL Features; Domain Features; Website Classification

1. Introduction

Phishing websites use deceptive branding, domains, and interaction flows to induce users to disclose information or initiate unauthorized actions. A URL may be short-lived, delivered through multiple channels, hidden behind a redirector, or constructed to resemble a familiar hostname. Manual inspection is therefore unreliable, especially when users must reason about subdomains, encoding, or long paths under time pressure. Surveys divide automated defenses into blacklist, heuristic, visual, content, and learning-based families, each with different latency and evidence requirements [1, 2].

Blacklist matching is inexpensive but necessarily depends on prior discovery. Page-content and visual methods can reveal forms, scripts, hyperlinks, or brand imitation, but they normally retrieve or render an untrusted resource. In

*Corresponding Author: Nithesh Naik (nitheshnayak@gmail.com)
Received: 02 June 2026; Revised: 09 July 2026; Accepted: 29 August 2026; Published: 30 September 2026
© 2026 The Author(s).
This is an open access article licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC 4.0).

contrast, lexical URL and lightweight host-structure features are available before navigation. Features such as length, delimiters, digit ratio, subdomain depth, suspicious terms, and character entropy can support fast screening without treating any single indicator as conclusive. Prior feature-selection work confirms that multiple URL manipulation strategies are measurable [3], while representation-learning work shows that raw character and token sequences also contain discriminative structure [4]. Deep representations may reduce manual engineering, but their training cost and lower transparency are not always justified for a baseline or resource-constrained component.

The objective is therefore: to design and evaluate a lightweight conceptual framework that classifies phishing and legitimate websites using interpretable URL and domain-related features and conventional machine-learning algorithms. Three questions guide the study: RQ1, which engineered features influence discrimination on the selected corpus; RQ2, how effectively do Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF) classify held-out URLs; and RQ3, which model offers the most useful balance of predictive metrics, interpretation, and computation. The contribution is fourfold in compact form: an original end-to-end framework; a reproducible 16-feature extractor that does not visit URLs; a leakage-aware comparison using verified data and measured outputs; and a restrained interpretation that separates prior evidence, the proposed design, and simulation findings.

2. Related Work

2.1. Blacklist and Rule-Based Detection

Blacklist and rule-based systems are attractive because a known indicator can be blocked quickly and a rule can be inspected. Their central weakness is coverage: an unseen or rapidly changing URL is not matched until observation and dissemination occur. The broader surveys in [1, 2] consequently describe learning-based and hybrid defenses as complements rather than replacements for reputation services. Rules also age when adversaries change naming conventions; conversely, aggressive rules can block legitimate URLs that share superficial patterns.

2.2. URL Lexical-Feature Approaches

Tharani and Arachchilage used information gain and chi-square selection on 48 features from 10,000 URLs to study URL-mimicry strategies [3]. URLNet instead learned character- and word-level representations from raw strings, reducing dependence on hand-designed predictors [4]. Hannousse and Yahiouche constructed a reproducible 11,430-URL benchmark with URL, content, and external-service features [5]. The systematic review by Safi and Singh shows that high reported scores coexist with heterogeneous datasets, feature sets, and validation procedures [6]. PhishHaven combined lexical signals and ensemble learning to address both human- and machine-generated phishing URLs [7].

2.3. Domain and Host-Based Approaches

Domain and host indicators can include hostname length, subdomain structure, IP-literal use, registration evidence, or DNS behavior. Extra-Trees meta-learners were evaluated on the established 11,055-instance phishing benchmark in [8]. Optimized stacking [9] and weighted soft voting [10] likewise show the appeal of ensemble methods on structured phishing features. Such systems can be accurate on their source benchmarks, but WHOIS, DNS, and certificate features may be unavailable, rate-limited, mutable, privacy-sensitive, or temporally inconsistent. The present framework therefore retains only host structure derivable from the URL string.

2.4. Machine-Learning-Based Phishing Classification

Hybrid evidence can improve coverage: Aljofey et al. combined URL character sequences, hyperlinks, and textual content [11]. Sánchez-Paniagua et al. evaluated URL analysis in the narrower setting of login URLs [12], whereas Alnemari and Alshammari concentrated on phishing-domain classification [13]. Kapan and Gunal compared multiple classifiers and feature configurations [14]; Abdul Samad et al. studied fine-tuned learning models [15]; and Yang et al. combined deep convolutional learning with RF [16]. Importantly, Ejaz et al. demonstrated performance deterioration across time-separated data and evaluated continual learning [17]. Recent alternatives include mutual-information-guided LR [18], knowledge-distilled ELECTRA [19], and BERT-derived features [20]. Rehman et al. used the PhiUSIIL corpus for an open machine-learning study [21], and Rawla et al. evaluated deep learning on the same corpus [22].

Table 1 summarizes representative, genuinely cited studies. Reported metrics are not pooled because their data, splits, labels, and evidence sources are not directly comparable.

Table 1: Comparison of representative phishing-detection studies

Ref.	Year	Feature category	Algorithms	Dataset type	Main contribution	Limitation relevant to this study
[3]	2020	URL/host engineered	Feature selection with ML	10,000 balanced URLs	Identified measurable URL-mimicry strategies	Earlier collection periods and a fixed engineered set
[5]	2021	URL, content, external	Classical ML benchmark	11,430 balanced webpages	Reproducible benchmark construction	Content/external queries exceed URL-only goals
[7]	2020	Lexical URL	Ensemble ML	Human- and AI-crafted URLs	Real-time ensemble design	Corpus construction complicates direct comparison
[8]	2020	URL, host, HTML/script	Extra-Trees meta-learners	11,055 structured records	Compared four Extra-Trees meta-learners	Uses a pre-extracted benchmark
[9]	2021	Structured website	Optimized stacking	Public benchmarks	Tuned stacked ensemble	Higher pipeline complexity
[10]	2021	Structured website	Weighted soft voting	Public benchmark	Combined heterogeneous learners	Reduced single-model transparency
[11]	2022	URL, hyperlinks, text	ML and deep learning	Two webpage datasets	Combined character, hyperlink, and text evidence	Requires webpage retrieval
[12]	2022	Lexical URL	ML and deep learning	Login-URL scenario	Focused evaluation on login URLs	Narrow landing-page scope
[13]	2023	Domain features	Multiple ML classifiers	Phishing/benign domains	Domain-focused feature analysis	Misses path/query behavior
[17]	2023	HTML deep features	Continual learning	Six time slices	Explicit temporal degradation test	Heavier extraction/retraining

2.5. Research Gap

Prior work establishes that URL, host, content, and learned representations are useful, but published scores often combine different corpora and validation assumptions. A compact, directly executable baseline remains valuable for teaching, auditing, ablation, and first-stage screening. This study deliberately asks a narrower question: what can be achieved with only a raw URL, 16 transparent features, three conventional classifiers, and no live network lookup? It also reports exact cleaning counts, a fixed split, timing boundaries, and threats that can inflate random-split performance.

3. Proposed Conceptual Framework

The proposed pipeline is shown in Figure 1. URL collection supplies labeled records from a scholarly corpus. Preprocessing removes missing strings, empty values, conflicts, and exact duplicates before splitting. Feature extraction parses the scheme, hostname, path, and query, then computes lexical and lightweight domain-structure indicators without contacting the destination. The feature vector is validated for finite numeric values. A stratified split preserves class proportions; model-specific preprocessing is learned only from training data. LR, DT, and RF produce phishing probabilities or class decisions. Evaluation combines error-sensitive metrics, elapsed times, a confusion matrix, and model influence analysis.

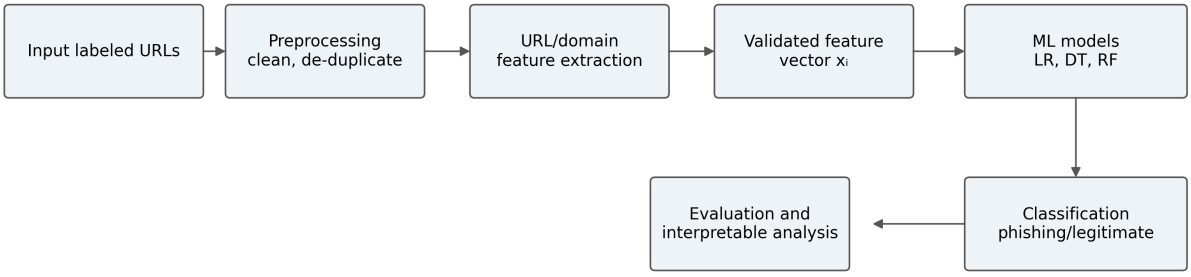


Figure 1: Proposed machine-learning framework for phishing website classification.

The framework is intentionally modular. A production system could add domain age, DNS, certificate, redirection,

content, or visual branches, but those branches would have separate caching, freshness, safety, and privacy controls. The current design does not infer legitimacy from HTTPS; it treats the scheme as one dataset-dependent signal among many.

3.1. Decision and Assurance Layer

A classifier score is not itself a deployment policy. The conceptual framework therefore separates model estimation from decision control. Two thresholds may define three outcomes: likely legitimate, likely phishing, and uncertain. The uncertain region can trigger a warning, delayed sandbox analysis, or analyst review rather than an irreversible block. Thresholds should be chosen using validation data and explicit costs for missed attacks, false alarms, review workload, and user interruption. Probability calibration should be measured before a score is presented as risk. Logging should retain the model version, feature extractor version, threshold, and reason codes while minimizing retention of complete URLs that may contain sensitive query values.

The assurance layer also monitors input distributions and outcomes. Shifts in length, entropy, suffix morphology, HTTPS prevalence, or suspicious terms can indicate benign ecosystem change, a new campaign, or collection drift. Monitoring should compare current feature distributions with the training baseline and initiate review rather than automatically accepting a retrained model. A candidate update should pass chronological, domain-grouped, and external tests, as well as regression checks on previously difficult legitimate URLs. This governance layer is conceptual in the present study; the simulation evaluates only the offline classifier stage.

4. URL and Domain Feature Engineering

For URL U, let L_u be its character length and N^d its number of numeric characters. The digit ratio is

$$R^d = \frac{N^d}{L_u}, \quad L_u > 0. \tag{1}$$

Let $p(c_i)$ denote the empirical probability of character c_i in U. Character entropy is

$$H(U) = -\sum_{i=1}^{m} p(c_i) \log_2 p(c_i). \tag{2}$$

Each URL U_i becomes the p-dimensional feature vector

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ip}], \quad p = 16. \tag{3}$$

For binary classification, the corresponding target is

$$y_i = \begin{cases} 1, & \text{phishing URL} \\ 0, & \text{legitimate URL} \end{cases} \tag{4}$$

Table 2 defines the feature set. “Special characters” are non-alphanumeric characters in the raw string. Suspicious-term count tests ten fixed case-insensitive substrings (login, verify, verification, account, secure, update, bank, signin, password, confirm). This transparent list is not a blacklist and should be revised under drift monitoring.

5. Research Methodology and Simulation Design

5.1. Dataset and Cleaning

The simulation uses PhiUSIIL_Phishing_URL_Dataset.csv, version 2, permanently identified by DOI 10.17632/shw-pxscxy2.2. An open conference study using this corpus is available in [21], and an open deep-learning study provides a further independent use in [22]. The downloaded file’s SHA-256 was verified as a236549c...7d06d1c6. It contained 235,795 records: 134,850 legitimate and 100,945 phishing. Only URL and label were read; precomputed content features, titles, and identifiers were excluded.

Table 2: Definitions of the extracted URL and domain features

Feature	Symbol	Definition	Security rationale	Type
URL length	L_u	Characters in the complete URL	Long strings can conceal host/path cues; not independently malicious	Integer
Hostname length	L_h	Characters in parsed hostname	Long hosts can contain deceptive labels or generated names	Integer
Path length	L_p	Characters in parsed path	Deep or verbose paths may obscure destination context	Integer
Query length	L_q	Characters after ? excluding fragment	Long parameters can carry tracking, redirection, or obfuscation	Integer
Dot count	$N_.$	Number of dot characters	Reflects hostname depth and dots outside the host	Integer
Hyphen count	N_-	Number of hyphens	Can occur in lookalike or compound labels	Integer
Digit count	N^d	Number of numeric characters	Generated paths and labels may contain many digits	Integer
Digit ratio	R^d	N^d/L_u	Normalizes digit use for URL length	Real
Special-character count	N^s	Number of non-alphanumeric characters	Captures delimiter and encoding complexity	Integer
Subdomain count	N_{sub}	Host labels beyond the final two; zero for IP literals	Deep subdomains can move trusted words from the registrable portion	Integer
IP-literal hostname	I_{IP}	One if host parses as IPv4/IPv6	Removes a human-readable domain identity	Binary
At-symbol present	$I_@$	One if @ occurs	User-information syntax can obscure the effective host	Binary
HTTPS scheme	I_{https}	One if scheme equals HTTPS	Informative in a corpus but never proves legitimacy	Binary
Suspicious-term count	N^t	Count of fixed credential/security terms	Credential and urgency vocabulary may occur in deceptive URLs	Integer
URL entropy	$H(U)$	Shannon character entropy	High diversity can indicate generated or obfuscated strings	Real
TLD length	L_{tld}	Characters in final hostname label	Coarse suffix morphology without a live registry query	Integer

There were no missing or empty URL/label values and no URLs with conflicting labels. Removing 425 exact duplicate URLs before splitting left 235,370 records: 134,850 legitimate and 100,520 phishing. The source convention (1 legitimate, 0 phishing) was explicitly inverted to match Equation (4). The 80:20 stratified split produced 188,296 training records and 47,074 test records. Table 3 records the remaining settings.

Table 3: Dataset and simulation configuration

Item	Verified setting
Source	PhiUSIIL v2; DOI 10.17632/shwpxscxy2.2
Raw records	235,795 (134,850 legitimate; 100,945 phishing)
Cleaning	425 exact URL duplicates removed; no missing/conflicting labels
Modeling records	235,370 (134,850 legitimate; 100,520 phishing)
Predictors	16 raw-URL-derived features; no live queries
Split	80:20 stratified; random state 42
Train/test counts	188,296 / 47,074
Models	LR; DT depth 10; RF 100 trees, depth 15
Scaling	StandardScaler fitted on training data for LR only
Selection rule	Highest test F1; ROC-AUC, recall, specificity as context

5.2. Models and Leakage Controls

LR used max_iter=1000; DT used max_depth=10; and RF used 100 estimators, max_depth=15, and parallel fitting. Each stochastic estimator used seed 42. These conservative values are proof-of-concept settings, not exhaustively optimized hyperparameters. LR was wrapped in a pipeline whose scaler was fitted only to training data; tree features were not scaled. Exact URLs were de-duplicated before splitting, labels and dataset identifiers were excluded from predictors, and the raw source's engineered features were not used. These controls prevent direct duplicate and preprocessing leakage, but they cannot exclude related domains or campaigns spanning both subsets.

5.3. Reproducibility Protocol

The source checksum binds the results to one published dataset version. The program writes a data-audit JSON file containing every removal count, final class counts, split counts, feature count, seed, and checksum. It also writes an

ordered metrics CSV and a feature-importance CSV, so table values can be regenerated rather than copied from a notebook state. Model ranking is deterministic for the fixed software environment and seed, apart from elapsed time and possible low-level numerical differences across platforms. Figure scripts read the in-memory executed results; they do not embed scores as plotting constants. This design makes discrepancies traceable to the input file, dependency versions, or runtime rather than to an undocumented manual step.

No cross-validation or hyperparameter search was performed, because the purpose is to establish a comprehensible baseline rather than estimate the best attainable score. Test results were inspected only after the three predeclared configurations were fitted. In a confirmatory study, a training-only cross-validation loop or a separate validation partition would choose features, thresholds, and hyperparameters, leaving the test set for a single final estimate.

6. Evaluation Metrics

Let TP and TN denote correct phishing and legitimate decisions, with FP a legitimate URL flagged as phishing and FN a missed phishing URL. The primary measures are

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5}$$

$$\text{Precision} = \frac{TP}{TP + FP}; \quad \text{Recall} = \frac{TP}{TP + FN}. \tag{6}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{7}$$

$$\text{Specificity} = \frac{TN}{TN + FP}. \tag{8}$$

ROC-AUC summarizes ranking across thresholds using predicted phishing probabilities. Recall matters because false negatives expose users to harmful pages; precision and specificity matter because excessive false alarms can interrupt legitimate work and erode trust. The simulation also records a confusion matrix, model-fit wall time, and prediction wall time. Timings exclude CSV loading and feature extraction and should be treated as environment-specific measurements rather than hardware-independent benchmarks.

7. Python Simulation Code

The complete executable program is supplied as run_simulation.py with pinned requirements and run instructions. It validates the required columns and SHA-256, maps labels, removes duplicates, extracts all features using urllib.parse, re, ipaddress, and collections, fits scikit-learn pipelines, writes CSV/JSON audit outputs, and generates vector PDF plus 300-dpi PNG figures. It never requests, visits, or executes a listed URL. Exceptions identify a missing file, invalid schema, nonbinary label, checksum mismatch, or nonfinite feature value. The source file, rather than manually transcribed pseudocode, is the authoritative listing.

8. Results and Discussion

8.1. Model Comparison

Table 4 reports only the outputs obtained by executing the supplied program on the verified file. RF ranked first by F1 and ROC-AUC; DT was close, while LR remained strong with the lowest-complexity decision function. Accuracy alone would hide the error asymmetry. RF missed 173 of 20,104 phishing test URLs and incorrectly flagged 18 of 26,970 legitimate URLs, giving the confusion matrix in Figure 2. LR produced fewer false positives (11) but more false negatives (280). Thus, RF's advantage arises mainly from higher phishing recall, at a small specificity cost.

Figure 3 compares precision, recall, F1, specificity, and ROC-AUC on a shared scale. RF required approximately five times the DT fitting time and four times the LR prediction time in this run, but all three model-only times were below one second for fitting and 0.1 seconds for the test-set prediction. These measurements do not include the Python-level feature loop; a deployed implementation should profile the complete pipeline and may vectorize or compile feature extraction.

Table 4: Executed simulation results on the stratified test set

Model	Accuracy	Precision	Recall	F1-score	Specificity	ROC-AUC	Train time (s)	Predict time (s)
Logistic Regression	0.993818	0.999445	0.986072	0.992714	0.999592	0.996530	0.427	0.023
Decision Tree	0.995730	0.999398	0.990599	0.994979	0.999555	0.996175	0.193	0.005
Random Forest	0.995943	0.999098	0.991395	0.995231	0.999333	0.998037	0.962	0.098

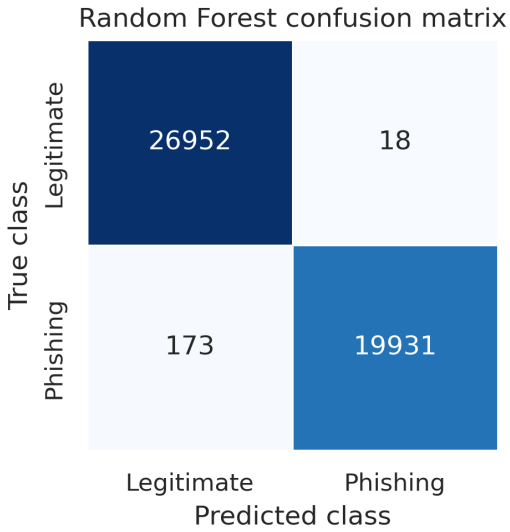


Figure 2: Confusion matrix of the best-performing classifier (Random Forest).

8.2. Interpretable Feature Analysis

RF's impurity-based importance in Figure 4 ranked HTTPS-scheme presence (0.4247), path length (0.2491), digit ratio (0.0842), special-character count (0.0662), and digit count (0.0579) highest. This ordering is descriptive of the fitted RF and selected random split. It does not imply causal security rules: HTTPS is widely available to both benign and malicious operators, and tree importance can favor variables that support many split points or correlate with other predictors. The result instead suggests that scheme, path morphology, and numeric/symbol composition jointly encode strong corpus-specific separation. The low importance of IP-literal and at-symbol indicators may reflect rarity, redundancy, or both; rare but operationally meaningful patterns should not be removed solely from this ranking.

The nonlinear trees can capture interactions such as a long path combined with particular character ratios. LR instead imposes a linear log-odds boundary after standardization, explaining part of its recall gap. DT captures nonlinear thresholds but a single depth-limited structure has less variance reduction than RF. Operationally, LR offers the simplest coefficient-level audit, DT offers rule paths, and RF offers the best tested discrimination with moderate interpretability through global importance. A threshold policy should still be selected on a separate validation set with costs and referral capacity, not fixed from this test comparison.

8.3. Error and Cost Interpretation

The 173 RF false negatives are the most security-sensitive errors because the classifier would pass phishing URLs at the default threshold. They can arise when a phishing page is hosted under an otherwise ordinary domain, when its path is short and natural, or when the URL lacks the patterns emphasized by the corpus. The 18 false positives may include legitimate authentication, account-management, or campaign-tracking URLs with long numeric or symbolic paths. These explanations are hypotheses consistent with the feature design; the study did not visit URLs or conduct a content-level forensic audit, so it does not assign causes to individual records.

A practical deployment would rarely use a single threshold for every context. Email gateways may tolerate a larger referral set than interactive browsing, whereas an automated takedown workflow needs stronger corroboration. For this reason, the measured confusion matrix is best interpreted as one operating point. ROC-AUC indicates strong ranking on the split, but it does not specify the alert burden under a deployment base rate, and a high precision on this corpus may fall when benign traffic greatly outnumbers phishing traffic. Prospective evaluation should therefore report precision-recall curves, calibration, workload at selected thresholds, and errors grouped by registrable domain and time.

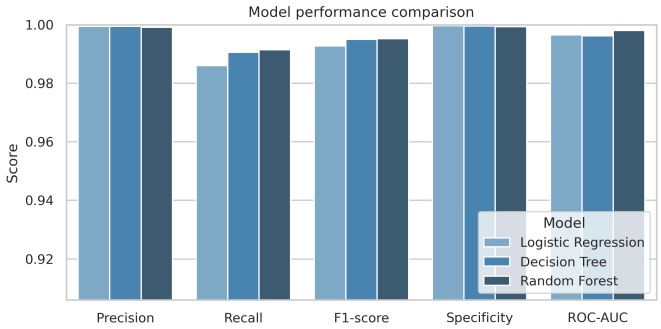


Figure 3: Comparison of classification metrics across the evaluated models.

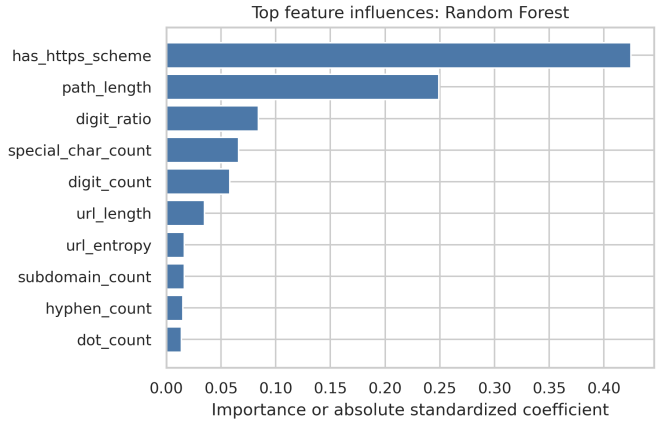


Figure 4: Ten largest impurity-based feature importances for Random Forest.

9. Limitations and Threats to Validity

First, PhiUSIIL reflects a particular collection process and period; its class priors and lexical cues may not represent a deployment stream. Exact duplicates were removed, but related hosts, paths, brands, or campaign templates can remain across a random split. This can overestimate generalization compared with domain-grouped, campaign-grouped, or chronological tests. Second, the moderate class imbalance favors legitimate records, although stratification preserves the ratio and class-specific metrics are reported. Third, phishing campaigns are short-lived and concept drift changes terms, hosting, certificates, and URL morphology; temporal degradation has been demonstrated in prior work [17].

Fourth, the features do not decode every obfuscation, follow redirects, resolve short links, normalize internationalized domains, or analyze character confusability. They omit domain age, DNS, TLS certificates, registration, webpage HTML, forms, scripts, screenshots, and behavior. Compromised legitimate domains and carefully minimal phishing paths can therefore evade lexical detection, while long legitimate authentication or tracking URLs can trigger false alarms. Fifth, the study evaluates one split and three conservative models without confidence intervals, cross-dataset validation, calibration assessment, adversarial testing, or deployment telemetry. The strong values demonstrate feasibility on the selected dataset; they do not establish universal, production-grade performance.

10. Ethical and Practical Considerations

This work is defensive. Dataset processing must treat URLs as potentially hazardous strings: analysts should not click them, request their content, render them, or execute associated code. Sharing should respect repository terms and should avoid exposing active malicious links in prose or logs. URL collections can also contain tokens, usernames, or other sensitive parameters, so access, retention, and output redaction deserve review. In deployment, a model score should be one signal within layered controls. High-impact blocking, account action, or user attribution should allow human review, documented overrides, and auditable error handling.

11. Conclusion and Future Work

This paper presented a lightweight conceptual framework and reproducible simulation for phishing-website classification using 16 URL and domain-structure features. On a de-duplicated random split of PhiUSIIL v2, all three conventional classifiers performed strongly; RF provided the best tested balance by F1 and ROC-AUC, while LR offered a simpler decision function and fewer false positives. The results show that inexpensive, interpretable URL morphology can support a useful first-stage screen, but they remain bounded by dataset and validation assumptions.

Future work should use chronological and registrable-domain-grouped splits, external corpora, calibration, drift monitoring, and online or continual updating. Character-level representations can be compared with engineered features under equal latency budgets. Carefully cached DNS, certificate, redirection-chain, webpage, and visual evidence could be added as optional branches. Explainable local decisions, adversarial URL perturbation tests, referral policies, and browser or email-security prototypes should be evaluated with human-centered false-alarm costs and safe isolation.

Author Contributions

Balakrishna S. Maddodi (B.S.M): Conceptualization, Methodology, Investigation, Formal analysis, Data curation, Writing – original draft; Nithesh Naik (N.N): Conceptualization, Methodology, Validation, Supervision, Writing – review & editing, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The data that support the findings of this study are available from the corresponding author, N.N, upon reasonable request.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

While preparing this manuscript, the authors made use of Paperpal and Grammarly to improve linguistic quality, grammar, readability, and textual presentation. Mendeley Cite was used for managing citations, organizing references, and generating the bibliography. Google Colab was utilized for computational tasks, including code execution, experimentation, and analysis. Additional AI-based tools were selectively employed to assist with preliminary drafting and enhance the efficiency of the writing process. All AI-assisted outputs were thoroughly reviewed, edited, and verified by the authors. The authors therefore assume full responsibility for the manuscript's accuracy, originality, scientific interpretation, and final content.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] L. Tang and Q. H. Mahmoud, 'A survey of machine learning-based solutions for phishing website detection,' Mach. Learn. Knowl. Extr., vol. 3, no. 3, pp. 672–694, 2021, doi: 10.3390/make3030034.

[2] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, 'A survey of intelligent detection designs of HTML URL phishing attacks,' IEEE Access, vol. 11, pp. 6421–6443, 2023, doi: 10.1109/ACCESS.2023.3237798.

[3] J. S. Tharani and N. A. G. Arachchilage, 'Understanding phishers' strategies of mimicking uniform resource locators to leverage phishing attacks: A machine learning approach,' Security and Privacy, vol. 3, no. 5, Art. no. e120, 2020, doi: 10.1002/spy2.120.

[4] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, 'URLNet: Learning a URL representation with deep learning for malicious URL detection,' arXiv:1802.03162, 2018.

[5] A. Hannousse and S. Yahiouche, 'Towards benchmark datasets for machine learning based website phishing detection: An experimental study,' Eng. Appl. Artif. Intell., vol. 104, Art. no. 104347, 2021, doi: 10.1016/j.engappai.2021.104347.

[6] A. Safi and S. Singh, 'A systematic literature review on phishing website detection techniques,' J. King Saud Univ. Comput. Inf. Sci., vol. 35, pp. 590–611, 2023, doi: 10.1016/j.jksuci.2023.01.004.

[7] M. Sameen, K. Han, and S. O. Hwang, 'PhishHaven—An efficient real-time AI phishing URLs detection system,' IEEE Access, vol. 8, pp. 83425–83443, 2020, doi: 10.1109/ACCESS.2020.2991403.

[8] Y. A. Alsariera, V. E. Adeyemo, A. O. Balogun, and A. K. Alazzawi, 'AI meta-learners and Extra-Trees algorithm for the detection of phishing websites,' IEEE Access, vol. 8, pp. 142532–142542, 2020, doi: 10.1109/ACCESS.2020.3013699.

[9] M. Al-Sarem et al., 'An optimized stacking ensemble model for phishing websites detection,' Electronics, vol. 10, no. 11, Art. no. 1285, 2021, doi: 10.3390/electronics10111285.

[10] A. Taha, 'Intelligent ensemble learning approach for phishing website detection based on weighted soft voting,' Mathematics, vol. 9, no. 21, Art. no. 2799, 2021, doi: 10.3390/math9212799.

[11] A. Aljofey et al., 'An effective detection approach for phishing websites using URL and HTML features,' Sci. Rep., vol. 12, Art. no. 8842, 2022, doi: 10.1038/s41598-022-10841-5.

[12] M. Sánchez-Paniagua, E. F. Fernández, E. Alegre, W. Al-Nabki, and V. González-Castro, 'Phishing URL detection: A real-case scenario through login URLs,' IEEE Access, vol. 10, pp. 42949–42960, 2022, doi: 10.1109/ACCESS.2022.3168681.

[13] S. Alnemari and M. Alshammari, 'Detecting phishing domains using machine learning,' Appl. Sci., vol. 13, no. 8, Art. no. 4649, 2023, doi: 10.3390/app13084649.

[14] S. Kapan and E. S. Gunal, 'Improved phishing attack detection with machine learning: A comprehensive evaluation of classifiers and features,' Appl. Sci., vol. 13, no. 24, Art. no. 13269, 2023, doi: 10.3390/app132413269.

[15] S. R. Abdul Samad et al., 'Analysis of the performance impact of fine-tuned machine learning model for phishing URL detection,' Electronics, vol. 12, no. 7, Art. no. 1642, 2023, doi: 10.3390/electronics12071642.

[16] R. Yang, K. Zheng, B. Wu, C. Wu, and X. Wang, 'Phishing website detection based on deep convolutional neural network and Random Forest ensemble learning,' Sensors, vol. 21, no. 24, Art. no. 8281, 2021, doi: 10.3390/s21248281.

[17] A. Ejaz, A. N. Mian, and S. Manzoor, 'Life-long phishing attack detection using continual learning,' Sci. Rep., vol. 13, Art. no. 11488, 2023, doi: 10.1038/s41598-023-37552-9.

[18] V. Vajrobol, B. B. Gupta, and A. Gaurav, 'Mutual information based logistic regression for phishing URL detection,' Cyber Security Appl., vol. 2, Art. no. 100044, 2024, doi: 10.1016/j.csa.2024.100044.

[19] K. S. Jishnu and B. Arthi, 'Real-time phishing URL detection framework using knowledge distilled ELECTRA,' Automatika, vol. 65, no. 4, pp. 1621–1639, 2024, doi: 10.1080/00051144.2024.2415797.

[20] M. Elsadig et al., 'Intelligent deep machine learning cyber phishing URL detection based on BERT features extraction,' Electronics, vol. 11, no. 22, Art. no. 3647, 2022, doi: 10.3390/electronics11223647.

[21] A. U. Rehman, I. Imtiaz, S. Javaid, and M. Muslih, 'Real-time phishing URL detection using machine learning,' Eng. Proc., vol. 107, no. 1, Art. no. 108, 2025, doi: 10.3390/engproc2025107108.

[22] A. Rawla, S. Singh, M. Daniyal, and P. Dubey, 'Detection of phishing attacks in PhiUSIIL dataset using deep learning,' Procedia Comput. Sci., vol. 259, pp. 543–552, 2025, doi: 10.1016/j.procs.2025.04.003.