© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

SmartPhish: A Confidence-Gated Two-Tier Learning Framework for Latency-Aware Phishing Website Detection

PRITI MISHRA
Department of Computer Science and Engineering (AI & ML), Sai Vidya Institute of Technology (affiliated to Visvesvaraya Technological University, Belagavi), Bengaluru, India

Abstract- ***Phishing is the most prevalent first-access point for credential theft and financial fraud on the Internet, and machine-learning detectors are the preferred defence. The accuracy figures reported in the literature, however, almost always come from feature vectors that tacitly assume that third-party infrastructure is always available: WHOIS registration databases, recursive DNS, TLS validators and search-engine index queries. These take hundreds of milliseconds per URL, disclose the browsing history of the user to outside parties, and do not work at all on a freshly registered phishing domain. This paper takes up a question that has received little attention in the literature on accuracy: how much of the reported accuracy actually requires the network? We provide an empirical answer on a public set of 88,647 labelled URLs described by 111 attributes, which we curate to 87,209 unique instances and 98 informative attributes. We first divide the feature space into 85 purely lexical attributes, computable from the URL string itself in 17.8 µs on a single core, and 13 external attributes, each requiring a query to a remote service. A comparative study of nine classifiers shows that histogram-based gradient boosting (HGB) is the best core learner, with an accuracy of 97.00%, an F1 score of 95.70% and an AUC of 99.56%. We then show that the lexical group alone reaches 93.50% accuracy while the external group alone reaches only 87.70%, and that the residual value of the external group is concentrated in a small set of hard-to-classify URLs that the system can identify in advance. We therefore propose a two-tier confidence-gated cascade in which a lexical-only model classifies every URL and a full-feature model is consulted only when the posterior of the lexical model falls inside an uncertainty band. Querying external services for only 19.72% of URLs, SmartPhish attains 96.60% accuracy and 95.10% F1 and cuts the expected end-to-end latency from 921 ms to 182 ms, a 5.1× improvement. All results are reproducible from public data and open-source software.***

Index Terms- ***cascade classifier, feature engineering, gradient boosting, latency-aware machine learning, Phishing detection, web security***

I. INTRODUCTION

Phishing is a social-engineering attack in which a web page is built to impersonate a trusted brand in order to steal credentials, payment information or session tokens. It is the most frequently reported category of cybercrime, and its economics favour the attacker: a phishing site is cheap to create and operate, is typically online for only a few hours, and is often not picked up by blacklists in time. Blacklist-based protection remains the foundation of much of the browser ecosystem, and it is by construction reactive: a URL must be reported, verified and published before any user is protected. The window between deployment and blacklisting is exactly the window in which the attack succeeds.

Machine learning offers a proactive alternative. A classifier learns the statistical regularities that separate phishing infrastructure from legitimate infrastructure and generalises to sites it has never encountered. A large body of work follows this path, and reported accuracies above 95% are now commonplace [1–4]. Progress in this field is usually expressed as an improvement in accuracy over a competing technique on a common test set.

This paper draws attention to the fact that accuracy is only one component of a figure of merit for a detector intended to run online, in the milliseconds between a user clicking a link and the browser rendering a page. Almost all high-scoring feature sets in the literature combine two kinds of attribute whose deployment costs differ by orders of magnitude.

Lexical attributes. Functions of the URL string itself: length, character counts, and the structure of the domain, path, filename and query. They are computed locally and deterministically, in microseconds, with no network traffic and no disclosure of the user's intended destination.

External attributes. Properties of the infrastructure behind the URL, such as domain age (WHOIS), time-to-live and resolved-address counts (DNS), mail- and name-server counts, TLS certificate validity, redirect chains, and whether the URL or domain appears in a search-engine index. Each of these requires at least one network round trip.

The difference is not superficial. A network lookup takes three to four orders of magnitude longer than lexical extraction, reveals the user's intended destination to a WHOIS registrar or a search engine, and is least informative exactly when it matters most: a domain registered minutes ago has no meaningful WHOIS history and no presence in any index. A detector that consults the network for every URL

therefore pays a large, unconditional cost for a benefit that is only occasional.

Contributions. This paper contributes to the literature in the following ways.

- We build a reproducible experimental pipeline over a public set of 88,647 labelled URLs, removing 1,438 duplicate URLs and 13 constant attributes whose retention would optimistically inflate every reported figure, and we state the exact preprocessing recipe (Section 3).
- We partition the 111 attributes in a principled way into 85 lexical and 13 external features, and quantify, for the first time on this corpus, the marginal detection value of the external group (Section 6.2).
- We report accuracy, Matthews correlation, false-positive rate, training cost and inference cost per URL for nine classifiers under an identical five-fold cross-validation protocol and a common hold-out split, and we establish which differences are statistically significant using McNemar tests (Sections 6.1 and 6.6).
- We propose SmartPhish, a confidence-gated two-tier cascade whose acquisition cost is data-dependent rather than constant, and we characterise the accuracy–latency trade-off across the whole range of the gate parameter (Sections 4 and 6.4).
- We show that a compact 30-feature model retains 99.6% of the full-model accuracy, which makes the approach practical for a browser extension with a limited memory budget (Section 6.3).

The remainder of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the dataset and the feature taxonomy. Section 4 presents the SmartPhish architecture. Section 5 states the experimental procedure, Section 6 reports and discusses the results, Section 7 examines threats to validity, and Section 8 concludes.

II. RELATED WORK

Heuristic and content-based detection. Early automated defences inspected the content of a suspect page after rendering it. CANTINA [5] extracted the highest-weighted TF–IDF terms from a page and queried a search engine, on the assumption that a legitimate page ranks well for its own terms; CANTINA+ [6] added HTML structure, URL features and a classification layer. PhishWHO [7] extended the identity-keyword idea with a target-identification stage. Content-based methods are informative but expensive: they must fetch and parse the page, which is slow and hazardous, and they can be defeated by pages that load their content dynamically through JavaScript or serve benign content to non-victim IP ranges.

URL-based statistical detection. A parallel line of work classifies the URL alone. Ma et al. showed that malicious URLs can be classified online with high accuracy from lexical and host-based features [8]. PhishStorm [9] introduced intra-URL relatedness measures derived from search-engine statistics. Sahingoz et al. [2] compared seven classifiers over NLP-based URL features and found Random Forest best, at 97.98% accuracy. Rao and Pais [3] combined URL, hyperlink and third-party features in a heuristic-plus-learning system. Our work belongs to this tradition, but treats the cost asymmetry inside the feature vector as the object of study rather than an implementation detail.

Deep learning approaches. URLNet [10] learns character- and word-level convolutional representations of raw URLs without manual feature engineering. Bahnsen et al. [11] applied recurrent networks to the raw URL string, and Aljofey et al. [12] used a character-level CNN. Such models are attractive because they need no external lookups at all, but they carry substantially higher inference budgets and are harder to audit than tree ensembles — a consideration for any security control that must explain why it blocked a user.

Ensembles and feature selection. The dominant techniques in the tabular phishing-detection literature are random forests [13] and gradient-boosted trees [14–16], which generally outperform kernel machines and shallow networks on standard phishing benchmarks [1, 17]. Feature-selection studies [18, 19] indicate that a small subset of attributes carries most of the discriminative information, which motivates Section 6.3. The dataset used here is an order of magnitude larger than the widely used UCI corpus [1]: 88,647 instances and 111 attributes, published by Vrbančič et al. [20].

Cascaded and cost-sensitive inference. Cascaded architectures, in which easy inputs are settled by a cheap model and hard inputs are forwarded to an expensive one, are classic in computer vision [21] and have re-emerged under the name of early-exit networks [22]. To the best of our knowledge, the idea has not previously been applied to phishing detection with the aim of reducing the cost of acquiring features over the network, which is the contribution of this paper.

III. DATA SET AND FEATURE TAXONOMY

A. Corpus

We use the public phishing-website set of Vrbančič et al. [20], released under an open licence. The corpus contains 88,647 URLs, each described by 111 numeric

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

features and a binary label; phishing URLs were drawn from PhishTank and legitimate URLs were sampled from the Alexa top-10-million list and Common Crawl. It was selected for three reasons: (1) it is roughly eight times larger than the smaller UCI corpus [1]; (2) its attributes are raw counts rather than pre-discretised −1/0/1 codes; and (3) the attribute names state unambiguously which URL component each count is derived from, which is what makes our lexical/external partition possible.

B. Curation

Two preprocessing steps materially affect the validity of any result on this corpus; both are applied.

Deduplication. The raw file contains 1,438 exactly duplicated rows. Left in place, these leak between the training and test partitions and optimistically bias every reported metric. We remove them, leaving 87,209 unique instances, of which 30,497 are phishing (34.97%) and 56,712 are legitimate.

Removal of degenerate features. Thirteen attributes are constant at zero for every instance: the counts of the slash, question-mark, equals, ampersand, exclamation, space, tilde, comma, plus, asterisk, hash, dollar and percent characters within the domain component — unsurprisingly, since these characters are not legal in a hostname. They carry no information and are dropped, leaving 98 attributes.

A further property of the corpus is that −1 is a sentinel value, occupying 39.18% of all cells and signifying either "not applicable" (for instance, directory-related counts for a URL with no directory component) or "lookup failed" (for instance, a WHOIS query that returned no registration date). We do not impute the sentinel for the tree ensembles; we retain it, because the event that a lookup fails is itself predictive.

C. Lexical versus external features

The 98 retained attributes are partitioned by acquisition cost.

The lexical group (85 attributes) comprises every attribute derivable by parsing the URL string: counts of each of the 17 special characters over the whole URL and over the domain, directory, filename and query components; the lengths of those components; the number of vowels in the domain; whether the domain is an IP address; whether the domain contains a client- or server-related keyword; whether an e-mail address is embedded in the URL; the number of top-level domains appearing in the query string; and whether the domain belongs to a known URL shortener. All of these are pure functions of a string.

The external group (13 attributes) consists of time_response, domain_spf, asn_ip, time_domain_activation, time_domain_expiration, qty_ip_resolved, qty_nameservers, qty_mx_servers, ttl_hostname, tls_ssl_certificate, qty_redirects, url_google_index and domain_google_index. Each requires at least one query to DNS, WHOIS, an SPF record, a TLS handshake, an HTTP fetch or a search engine.

To quantify the asymmetry concretely, we implemented a reference lexical extractor in pure Python and timed it over 20,000 synthetic URLs on a single core: it requires 17.8 µs per URL. The dataset itself records the measured HTTP response time of each host in milliseconds in the field time_response, whose mean over successful lookups is 921 ms. A single network round trip therefore costs about 5×10^4 times more than the entire lexical feature extraction, and several round trips are needed to complete the external feature vector. This ratio motivates the method that follows.

IV. THE SMARTPHISH FRAMEWORK

A. Design rationale

Let $x \in X$ be a URL, let $\varphi_{\mathrm{L}}(x) \in \mathbb{R}^{85}$ be its lexical feature vector, and let $\varphi_{\mathrm{F}}(x) \in \mathbb{R}^{98}$ be the full vector of lexical and external attributes. Write c_{L} and c_{F} for the acquisition costs of the two vectors, with $\frac{c_{\mathrm{F}}}{c_{\mathrm{L}}} \approx 5 \times 10^4$ as established above. A conventional detector that evaluates a single model $h_{\mathrm{F}}(x)$ pays c_{F} on every call.

The key empirical finding, quantified in Section 6.2, is that the lexical model is not uniformly inferior to the full model: it is inferior on some inputs, and, crucially, it knows which ones. A well-calibrated probabilistic classifier places its posteriors near 0 or 1 on inputs it is confident about and near 0.5 on inputs it is not. If the errors concentrate inside that uncertain band, then it is worth spending the expensive budget only within the band, because most of the accuracy can be recovered at a fraction of the cost.

B. Architecture

SmartPhish instantiates this idea as the two-tier cascade of Figure 1. Tier 1 is a histogram-based gradient-boosting classifier h_{L} trained on φ_{L} alone. Given a gate parameter $\tau \in [0.5, 1]$, a URL is *resolved* at Tier 1 if its Tier-1 posterior lies outside the uncertainty band, and *escalated* otherwise. Escalated URLs trigger acquisition of the external attributes and are re-classified by Tier 2, a model h_{F} trained on the full vector. The final posterior is

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

$$\hat{p}(x) = \begin{cases} h_{\mathrm{L}}(\varphi_{\mathrm{L}}(x)), & \text{if } h_{\mathrm{L}}(\varphi_{\mathrm{L}}(x)) \notin (1-\tau, \tau), \\ h_{\mathrm{F}}(\varphi_{\mathrm{F}}(x)), & \text{otherwise.} \end{cases} \quad (1)$$

and the decision rule is $\hat{y}(x) = \mathbb{1}[\hat{p}(x) \geq 0.5]$. Writing $e(\tau) = \Pr[h_{\mathrm{L}}(\varphi_{\mathrm{L}}(x)) \in (1-\tau, \tau)]$ for the escalation rate — the fraction of URLs for which the network must be consulted — the expected per-URL cost is

$$\mathrm{E}[C(\tau)] = c_{\mathrm{L}} + t_1 + e(\tau) \cdot (c_{\mathrm{F}} + t_2) \quad (2)$$

where t_1 and t_2 are the two inference times. The extremes recover the degenerate designs: $\tau = 0.5$ gives an escalation rate of zero and a pure lexical detector, while $\tau = 1$ gives an escalation rate of one and the conventional always-on full detector. The interesting regime lies strictly between them.

C. Algorithm

Algorithm 1 states training and inference. Both tiers are trained on the same partition, so no additional labelled data is required, and the gate is selected on cross-validated training data, never on the test partition.

Algorithm 1. SmartPhish training and confidence-gated inference
Input: Training set D of n labelled URLs; lexical index set L; full index set F; candidate gates T; budget β on the escalation rate.
Output: Tier models hL, hF; gate τ*.
Training
1: hL ← FitHGB({(φ(xi)[L], yi)})
2: hF ← FitHGB({(φ(xi)[F], yi)})
3: for each τ ∈ T do
4: compute cascade predictions on held-out folds via Eq. (1)
5: record accuracy a(τ) and escalation rate e(τ)
6: end for
7: τ* ← argmax {a(τ) : τ ∈ T, e(τ) ≤ β}
Inference on a new URL x
1: p ← hL(φL(x)) (no network traffic)
2: if p ≤ 1 − τ* or p ≥ τ* then return 𝟙[p ≥ 0.5] (fast path)
3: else acquire external attributes, set p ← hF(φF(x)), and return 𝟙[p ≥ 0.5] (slow path)
4: end if

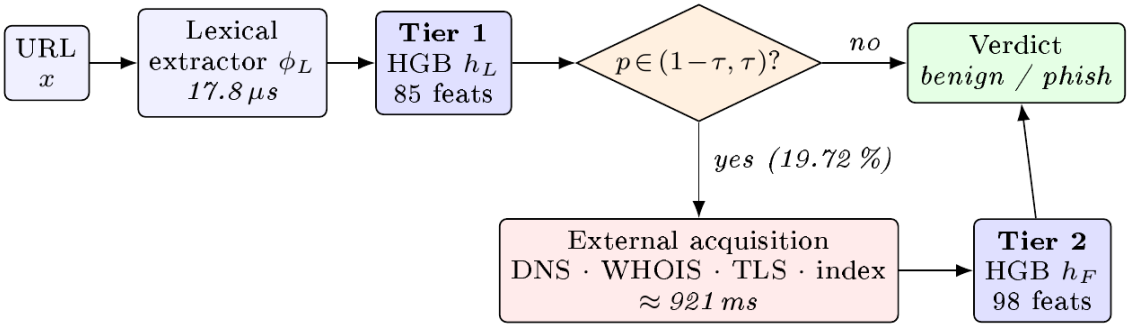


Figure 1. The SmartPhish confidence-gated cascade. The slow path (shaded) is taken only for URLs whose Tier-1 posterior falls inside the uncertainty band; at the selected gate τ = 0.90 this is 19.72% of traffic.

V. EXPERIMENTAL SETUP

Protocol. The curated corpus of 87,209 instances is split once, stratified by label, into 61,046 training instances (70%) and 26,163 test instances (30%) under a fixed random seed. All model selection, hyper-parameter choice and gate calibration use five-fold stratified cross-validation within the training partition only; the test partition is touched once, to produce the reported hold-out numbers.

Models. Nine classifiers are compared: Gaussian naive Bayes; ℓ2-regularised logistic regression; a linear SVM (C = 0.5, squared hinge, primal solver); k-nearest neighbours (k = 5, Euclidean); a CART decision tree grown to purity; a multilayer perceptron with two hidden layers (100 and 50 units, ReLU, Adam, early stopping); Random Forest (100 trees); Extra Trees (100 trees); and histogram-based gradient boosting (HGB; 300 boosting iterations, learning rate 0.1). Distance- and gradient-based learners are preceded by standardisation inside the cross-validation pipeline to prevent leakage. An RBF SVM was excluded because its $O(n^2)$ training cost is prohibitive at this sample size, a limitation noted explicitly in Section 7.

Metrics. We report accuracy, precision, recall, F1, Matthews correlation coefficient (MCC), ROC AUC and false-positive rate (FPR), taking phishing as the positive class. MCC is included because the corpus is imbalanced at roughly 35/65 and, unlike accuracy, MCC degrades sharply for a classifier that merely exploits the majority prior. FPR is singled out because, in deployment, a false positive blocks a legitimate site — the failure mode users notice and vendors fear.

Environment. All experiments ran single-threaded on one x86-64 core with 3 GiB of RAM under Python 3, scikit-learn 1.8 [23], NumPy 2.4 and pandas 3.0.

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

Restricting execution to a single core is deliberate: it makes the reported training and inference times a conservative, hardware-modest baseline.

VI. RESULTS AND DISCUSSION

A. Comparison of core classifiers

Table 1 reports all nine classifiers on the full 98-attribute vector. The ordering is stable between cross-validation and hold-out, and the cross-validation standard deviations are small (≤ 0.72 percentage points), so the ranking is not an artefact of a lucky split.

Table 1. Comparison of nine classifiers on the full 98-feature vector. Cross-validation accuracy is the mean ± standard deviation over five stratified folds of the training partition; all other columns are measured on the untouched hold-out partition of 26,163 URLs. Times are single-core.

Classifier	CV acc. (%)	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	MCC (%)	AUC (%)	Train (s)	Infer. (µs)
Gaussian Naive Bayes	78.31 ± 0.72	78.06	88.40	42.88	57.75	50.64	94.55	0.1	0.9
Logistic Regression	93.24 ± 0.17	93.42	90.03	91.29	90.65	85.58	97.97	0.9	0.7
Linear SVM	93.12 ± 0.15	93.30	89.29	91.87	90.56	85.39	97.94	3.4	0.4
k-NN (k = 5)	95.03 ± 0.13	95.21	93.70	92.52	93.11	89.44	98.06	0.1	464.2
Decision Tree (CART)	94.97 ± 0.20	95.02	92.84	92.94	92.89	89.06	94.55	0.9	0.2
MLP (100–50)	95.71 ± 0.17	95.84	93.76	94.38	94.07	90.86	99.18	6.1	1.9
Extra Trees	96.70 ± 0.13	96.91	95.72	95.43	95.58	93.21	99.48	6.8	16.6
Random Forest	96.75 ± 0.03	96.97	95.68	95.64	95.66	93.33	99.50	6.9	11.8
HGB (proposed core)	96.83 ± 0.10	97.00	95.98	95.42	95.70	93.40	99.56	9.3	17.3

Three observations follow. First, the tree ensembles form a clear top group: HGB, Random Forest and Extra Trees all exceed 96.9% hold-out accuracy and 99.4% AUC, well ahead of the MLP (95.84%) and far ahead of the linear models (≈ 93.3%). This is consistent with the wider tabular-learning literature and with the structure of the problem: the decision boundary involves many threshold interactions on count features, which axis-aligned splits capture efficiently and linear separators cannot.

Second, Gaussian naive Bayes collapses to 78.06% accuracy with 42.88% recall despite a respectable 94.55% AUC. The gap between AUC and accuracy is diagnostic: the model ranks instances tolerably, but its posteriors are badly calibrated, so the 0.5 threshold sits in the wrong place. The conditional-independence assumption is untenable here — counts such as qty_slash_url and directory_length are near-deterministic functions of one another.

Third, inference cost varies by more than three orders of magnitude across the top group. HGB needs 17.3 µs per URL while k-NN needs 464 µs, because the latter defers all work to prediction time. For an inline security control this alone disqualifies k-NN irrespective of its accuracy. We adopt HGB as the core learner for both cascade tiers: it is the most accurate model tested, its training cost (9.3 s) is modest, and its native handling of the −1 sentinel avoids an imputation step.

B. How much does the network buy? A feature-group ablation

Table 2 isolates the contribution of each feature group by retraining HGB on each in turn.

Table 2. Feature-group ablation with HGB. The lexical group requires no network traffic; the external group requires several remote lookups per URL.

Feature group	CV acc. (%)	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	MCC (%)	AUC (%)	FPR (%)
Lexical only (85)	93.42 ± 0.17	93.50	92.06	89.10	90.56	85.63	98.06	4.13
External only (13)	87.69 ± 0.15	87.70	83.68	80.52	82.07	72.74	94.44	8.45
All features (98)	96.83 ± 0.10	97.00	95.98	95.42	95.70	93.40	99.56	2.15

The lexical group alone reaches 93.50% accuracy and 98.06% AUC — that is, 96.4% of the full-model accuracy at zero network cost. The external group alone reaches only 87.70%, barely 12.7 points above

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

the majority-class baseline of 65.03%. Yet combining the groups adds 3.5 points over lexical-only, so the external attributes are genuinely complementary rather than redundant; they are simply far less informative per unit of cost than a naive reading of a feature-importance chart suggests.

Figure 2 and Table 3 refine the picture. The single most important attribute by mean decrease in impurity is directory_length, a lexical feature, followed immediately by time_domain_activation — domain age, the classic WHOIS signal, which alone accounts for 6.4% of total impurity reduction. Summed over the group, external attributes carry 21.2% of the total importance mass while constituting only 13.3% of the attributes, so they are indeed disproportionately informative individually. But importance mass measures contribution to a fitted model, not irreplaceable information: the ablation shows that when the external group is withheld, the tree ensemble recovers most of that signal from correlated lexical structure. The two measurements are not in conflict, and reading only the first is precisely the error this paper cautions against.

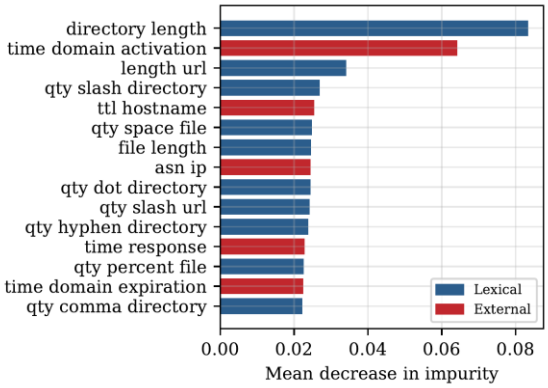


Figure 2. Fifteen highest-ranked attributes by mean decrease in impurity (Random Forest, 200 trees). Lexical attributes dominate the ranking by count; external attributes are individually strong but few.

Table 3. Fifteen highest-ranked attributes by mean decrease in impurity, with their mutual information with the label for comparison.

Rank	Attribute	Group	Impurity decrease	Mutual info.
1	directory_length	Lexical	0.0834	0.3781
2	time_domain_activation	External	0.0643	0.1943
3	length_url	Lexical	0.0342	0.2771
4	qty_slash_directory	Lexical	0.0270	0.3717
5	ttl_hostname	External	0.0255	0.0918
6	qty_space_file	Lexical	0.0248	0.3305
7	file_length	Lexical	0.0246	0.3476
8	asn_ip	External	0.0245	0.1601
9	qty_dot_directory	Lexical	0.0245	0.3485
10	qty_slash_url	Lexical	0.0242	0.3717
11	qty_hyphen_directory	Lexical	0.0239	0.3392
12	time_response	External	0.0229	0.0107
13	qty_percent_file	Lexical	0.0226	0.3267
14	time_domain_expiration	External	0.0224	0.0531
15	qty_comma_directory	Lexical	0.0223	0.3293

C. How many features are actually needed?

Table 4 and Figure 3 report HGB trained on the top-k attributes of the impurity ranking.

Table 4. Effect of the number of selected attributes k on detection quality.

k	CV accuracy (%)	Accuracy (%)	F1 (%)	AUC (%)
5	94.85 ± 0.10	94.90	92.76	98.88
10	95.94 ± 0.09	96.09	94.43	99.28
15	96.18 ± 0.08	96.36	94.80	99.35
20	96.14 ± 0.11	96.34	94.78	99.37
25	96.20 ± 0.07	96.38	94.84	99.37
30	96.57 ± 0.07	96.65	95.20	99.46
40	96.61 ± 0.14	96.82	95.44	99.51
50	96.80 ± 0.06	96.92	95.59	99.54
70	96.87 ± 0.12	97.01	95.72	99.55
98	96.83 ± 0.10	97.00	95.70	99.56

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

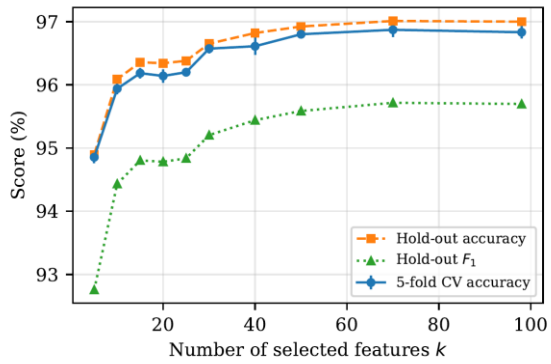


Figure 3. Detection quality as a function of the number of selected attributes k. Performance saturates near k = 30; error bars are one standard deviation over five cross-validation folds.

Just five attributes already yield 94.90% accuracy; ten yield 96.09%; thirty yield 96.65%, which is 99.6% of the full-model figure. Beyond k = 50 the curve is flat to within cross-validation noise, and k = 70 marginally exceeds the full set (97.01% versus 97.00%), a mild regularisation effect from discarding twenty-eight weak attributes. The practical consequence is that a deployed model can carry a third of the attributes at essentially no accuracy cost — relevant for a browser extension, where every retained attribute is code that must be written, maintained and audited.

D. The confidence-gated cascade

Table 5 and Figure 4 sweep the gate τ over its full range. The latency column instantiates Eq. (2) with the measured constants of Section 3: $c_L + t_1 = 28.0\ \mu s$ and $c_F + t_2 \approx 921\ ms$, the latter taken from the dataset's own recorded HTTP response times and therefore a conservative estimate, since a complete external vector needs a WHOIS query, DNS resolution, TLS validation and two index lookups in addition to the HTTP fetch.

Table 5. Accuracy–latency trade-off of the SmartPhish cascade as a function of the confidence gate τ. Row τ = 0.50 is the pure lexical detector and row τ = 1.00 is the conventional always-on full detector. The selected operating point is shown in bold.

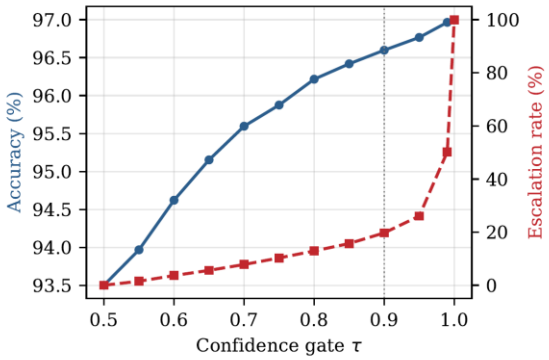


Figure 4. Accuracy (left axis) and escalation rate (right axis) against the confidence gate τ. The dotted line marks the selected operating point τ = 0.90, past which accuracy gains flatten while escalation grows steeply.

The trade-off is markedly convex. Escalating just 7.81% of URLs (τ = 0.70) recovers 2.10 of the 3.50 accuracy points that separate the lexical model from the full model. At τ = 0.90, escalating 19.72% of URLs yields 96.60% accuracy, 95.10% F1, 92.50% MCC and a 2.25% false-positive rate — within 0.40 accuracy points of the full model — while cutting expected latency from 921 ms to 182 ms, a 5.1× reduction, and cutting external queries by the same factor. Beyond τ = 0.95 the curve turns sharply: moving from τ = 0.95 to τ = 0.99 doubles the escalation rate from 26.04% to 50.24% to buy 0.20 accuracy points. We therefore select τ = 0.90, the knee of the curve.

Figure 5 shows the resulting confusion matrix. Of 26,163 test URLs, 8,642 phishing pages are caught and 507 are missed, while 383 legitimate pages out of 17,014 are incorrectly blocked.

The privacy dimension is worth stating separately, because it is not visible in any accuracy column. In the always-on design, every URL a user visits is disclosed to a WHOIS registrar and a search engine. Under SmartPhish that disclosure occurs for roughly one URL in five and, by construction, only for URLs the lexical model finds ambiguous. The cascade is thus a privacy-preserving mechanism as much as a latency-reducing one.

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/ire1723029

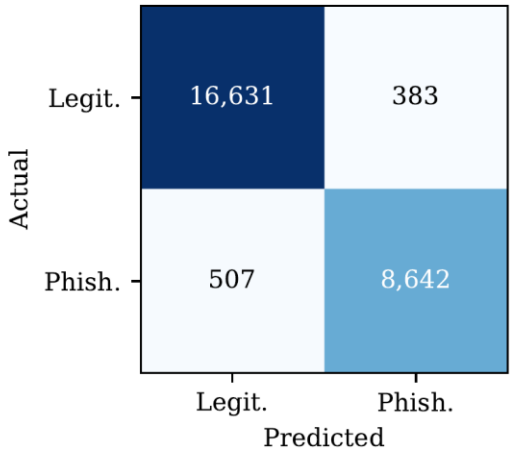


Figure 5. Confusion matrix of SmartPhish at τ = 0.90 on the 26,163-URL hold-out partition.

E. Operating at a low false-positive rate

Accuracy at the default 0.5 threshold is a poor guide to deployability, because a browser vendor will not tolerate blocking 2% of legitimate traffic. Constraining FPR to 1.00% by moving the decision threshold, the full-feature model detects 90.98% of phishing URLs and the cascade detects 89.05% — a gap of under two points. Figure 6 shows the two curves over the region FPR ≤ 10%, where they are close to indistinguishable above about 3% FPR. The cost advantage of the cascade is therefore not purchased at the price of the operating region that matters.

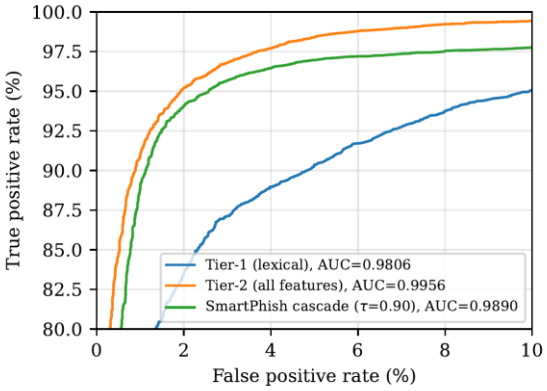


Figure 6. ROC curves in the operationally relevant low-FPR region. The cascade tracks the full-feature model closely while consulting the network for one URL in five.

F. Statistical significance

Because all models are evaluated on the same 26,163 test instances, paired McNemar tests on the discordant pairs are the appropriate instrument [24]. Table 6 reports n01 (instances the first model classifies correctly and the second does not), n10 (the converse), the continuity-corrected $\chi 2$ statistic and its p-value.

Table 6. Paired McNemar tests on the 26,163-URL hold-out partition. n01 counts instances correct for the first model only, n10 for the second only.

The advantage of HGB over the MLP, k-NN and logistic regression is significant at any conventional level ($p < 10{-}15$), but its advantage over Random Forest is not ($\chi 2 = 0.15$, $p = 0.70$), nor over Extra Trees ($p = 0.34$). We therefore make no claim that HGB is superior to Random Forest on this corpus; we select it for its lower training cost and native sentinel handling, and note that Random Forest would serve equally well as the cascade core. Reporting this negative result matters: differences of a few tenths of a point between strong ensembles are routinely presented as improvements in the phishing-detection literature without significance testing, and on this evidence such differences are noise.

The cascade at $\tau = 0.90$ is significantly worse than the full model ($\chi 2 = 63.25$, $p < 10{-}14$; it loses 138 instances and gains 33) and significantly better than the lexical-only model ($\chi 2 = 523.58$, $p < 10{-}15$; it gains 1,030 and loses 220). We state the first result plainly rather than obscuring it: the cascade does trade a small, statistically real amount of accuracy for a five-fold reduction in network cost. The contribution is the favourable exchange rate — 0.40 accuracy points for 80% of the lookups — not a free lunch.

VII. THREATS TO VALIDITY

Data source and drift. The corpus is a fixed time-snapshot. Absolute numbers should be read as the performance attainable on that snapshot rather than as the standing figures for any current phishing campaign, since phishing infrastructure evolves adversarially. The legitimate class is also drawn from popular, heavily accessed sites, which may make it easier to separate than the long tail of minor legitimate sites.

Feature-partition assumptions. Our lexical/external partition follows the semantics of the attribute names. Two attributes sit close to the boundary: url_shortened is a comparison against a small table of shortener domains, and domain_in_ip is a string test. We classify both as lexical; reclassifying them would change the group sizes by at most two attributes.

Latency model. We instantiate the external cost constant in Eq. (2) with the mean HTTP response time recorded in the dataset. Depending on network conditions, caching and rate limits, the effective cost may differ considerably in practice: a client that caches WHOIS results per domain would see a lower cost, and a client rate-limited by a search engine a higher one. The escalation rate is the more robust figure of merit, because it is measured directly and is

independent of any timing assumption, and we report it throughout alongside latency.

Excluded model families. An RBF-kernel SVM was not evaluated because of tractability at n = 61,046 on a single core, and large deep sequence models were not considered because they would introduce a change of representation in addition to a change of learner. Both are natural extensions.

Calibration. The gate assumes that Tier-1 posteriors are informative about correctness. Gradient-boosted trees are reasonably well calibrated out of the box, and the monotonic accuracy curve in τ shown in Figure 4 confirms that this holds here, although explicit calibration (isotonic or Platt scaling) might move the knee to a more favourable position. We do not explore this.

VIII. CONCLUSION AND FUTURE WORK

We revisited machine-learning phishing detection while accounting explicitly for the cost of acquiring each feature. On a curated public corpus of 87,209 URLs we found that combining 85 purely lexical attributes — extractable in 17.8 µs without touching the network — with 13 network-dependent ones attains 97.00% accuracy, 95.70% F1 and 99.56% AUC, whereas the 13 network-dependent attributes reach only 87.70% accuracy on their own. On that basis we proposed SmartPhish, a confidence-gated cascade in which the full-feature model is consulted only when the posterior of the cheaper lexical model is uncertain. At the selected gate the system attains 96.60% accuracy and 95.10% F1 while issuing external queries for 19.72% of URLs, lowering expected latency by a factor of 5.1 and reducing by the same factor the number of URLs disclosed to third parties. We also showed that 30 attributes suffice to reach 99.6% of the full accuracy, and we reported rather than concealed the finding that the strongest tree ensembles are not significantly different from one another.

Several directions follow. First, the gate is a global threshold; a learned, input-dependent deferral policy should outperform it. Second, per-attribute rather than per-group escalation would let the system acquire only the external attribute most likely to resolve a particular ambiguity, under a value-of-information criterion. Third, the framework should be tested against adversarial manipulation of lexical features, since an attacker familiar with the Tier-1 model could craft URLs that are confidently misclassified and therefore never escalated — the core security question raised by any cascade. Finally, a live browser-extension deployment would replace our analytical latency model with end-to-end measurements on a real network.

Reproducibility. The dataset is publicly available [20]; all experiments use open-source libraries with fixed random seeds, and the preprocessing, splits and hyper-parameters are stated in full in Sections 3 and 5.

REFERENCES

[1] Mohammad RM, Thabtah F, McCluskey L. Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications. 2014;25(2):443–458.

[2] Sahingoz OK, Buber E, Demir O, Diri B. Machine learning based phishing detection from URLs. Expert Systems with Applications. 2019;117:345–357.

[3] Rao RS, Pais AR. Detection of phishing websites using an efficient feature-based machine learning framework. Neural Computing and Applications. 2019;31(8):3851–3873.

[4] Jain AK, Gupta BB. Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems. 2018;68(4):687–700.

[5] Zhang Y, Hong JI, Cranor LF. CANTINA: a content-based approach to detecting phishing web sites. In: 16th International Conference on World Wide Web; 2007. p. 639–648.

[6] Xiang G, Hong J, Rose CP, Cranor L. CANTINA+: a feature-rich machine learning framework for detecting phishing web sites. ACM Transactions on Information and System Security. 2011;14(2):1–28.

[7] Tan CL, Chiew KL, Wong K, Sze SN. PhishWHO: phishing webpage detection via identity keywords extraction and target domain name finder. Decision Support Systems. 2016;88:18–27.

[8] Ma J, Saul LK, Savage S, Voelker GM. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2009. p. 1245–1254.

[9] Marchal S, François J, State R, Engel T. PhishStorm: detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management. 2014;11(4):458–471.

[10] Le H, Pham Q, Sahoo D, Hoi SCH. URLNet: learning a URL representation with deep learning for malicious URL detection. arXiv:1802.03162. 2018.

[11] Bahnsen AC, Bohorquez EC, Villegas S, Vargas J, González FA. Classifying phishing URLs using recurrent neural networks. In: APWG Symposium on Electronic Crime Research (eCrime). IEEE; 2017. p. 1–8.

[12] Aljofey A, Jiang Q, Qu Q, Huang M, Niyigena JP. An effective phishing detection model based on character level convolutional neural network from URL. Electronics. 2020;9(9):1514.

[13] Breiman L. Random forests. Machine Learning. 2001;45(1):5–32.

[14] Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of Statistics. 2001;29(5):1189–1232.

[15] Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785–794.

[16] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY. LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems; 2017. Vol. 30. p. 3146–3154.

[17] Korkmaz M, Sahingoz OK, Diri B. Detection of phishing websites by using machine learning based URL analysis. In: 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE; 2020. p. 1–7.

[18] Zouina M, Outtaj B. A novel lightweight URL phishing detection system using SVM and similarity index. Human-centric Computing and Information Sciences. 2017;7(1):17.

[19] Verma R, Das A. What's in a URL: fast feature extraction and malicious URL detection. In: 3rd ACM International Workshop on Security and Privacy Analytics; 2017. p. 55–63.

[20] Vrbančič G, Fister I Jr, Podgorelec V. Datasets for phishing websites detection. Data in Brief. 2020;33:106438.

[21] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2001. Vol. 1. p. 511–518.

[22] Teerapittayanon S, McDanel B, Kung HT. BranchyNet: fast inference via early exiting from deep neural networks. In: 23rd International Conference on Pattern Recognition (ICPR). IEEE; 2016. p. 2464–2469.

[23] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

[24] Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation. 1998;10(7):1895–1923.