Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# StackShield: Stacked Character-Level and Explainable URL Features for Phishing Detection in QR/NFC Systems

**Mustafa AL SAMARA**

malsamara@jbj.edu.qa

Joaan Bin Jassim Academy for Defence Studies

**Research Article**

**Keywords:** QR code security, NFC security, quishing, phishing URL detection, cross-dataset evaluation, explainable AI

**Posted Date:** November 2nd, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-11394102/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# StackShield: Stacked Character-Level and Explainable URL Features for Phishing Detection in QR/NFC Systems

**Abstract**

QR codes and NFC tags hide their destination URL until the user scans or taps, so phishing (quishing) must be detected from the URL alone, before any page loads. We show that the 97–99% accuracy reported for URL detectors does not hold under these scan-time conditions. On the UCI Phishing Websites dataset, accuracy falls from 97.2% to 73.7% when only URL-string features are used. In PhiUSIIL, every legitimate URL is an https://www. homepage, so a one-line rule reaches 99.1% accuracy; the same rule exceeds 99.6% on two further benchmarks, and a PhiUSIIL-trained model performs at chance level on four other datasets. We propose StackShield, a scan-time detector that stacks the out-of-fold score of a character n-gram model with 116 explainable lexical, structural, hosting and brand-impersonation features in an XGBoost classifier. On 166,542 PhreshPhish URLs collected after the training period, StackShield raises balanced accuracy from 89.1% to 96.8% (AUC 99.5%) and detects 94.4% of phishing at a 1% false-positive rate, versus 71.9% for the baseline. Detection of lookalike https://www. phishing rises from 66% to 85%, and of brand impersonation from 71% to 97%. A heterogeneous stacking ensemble adds no practical gain at 35 times the training cost. StackShield classifies a URL in 25 ms and explains its decisions with SHAP.

**Keywords:** QR code security, NFC security, quishing, phishing URL detection, cross-dataset evaluation, explainable AI

## 1 Introduction

Quick Response (QR) codes and Near Field Communication (NFC) tags, standardized in ISO/IEC 18004 [1] and ISO/IEC 14443 [2] respectively, have become everyday interfaces between physical and digital environments. They are used for mobile payments, digital menus, parking, event registration, contactless identification, and e-government services. Their appeal lies in speed and convenience: a single scan or tap opens a digital service. The same convenience creates a security problem, because the destination URL remains hidden until after the user has interacted.

Users typically scan first and verify later, relying on implicit trust in the physical setting in which a code or tag appears. Attackers exploit this by placing malicious

QR codes or NFC tags that lead to phishing pages, credential-harvesting forms, or malware downloads, an attack known as *quishing* when delivered through QR codes. User studies show how effective this is: in one study, 85% of participants followed URLs from publicly posted QR codes without inspecting them [3], and in another, 67% of participants who scanned a malicious QR code disclosed personal information on a counterfeit website [4]. Because the user never sees the URL before the page opens, protection must come from classifying the URL automatically at *scan time*, after it is decoded from the code or tag and before any connection to the destination is made.

Al-Mannai et al. [5] built a web-based pre-interaction platform that performs this check for URLs extracted from QR codes and NFC tags on Android devices. Its best classifier, XGBoost, reached 90.15% accuracy on a balanced dataset of 20,000 URLs described by 30 engineered features. Published phishing URL detectors often report 97–99% accuracy on public benchmarks, which would suggest that far better results are easily attainable. This paper shows that this suggestion is misleading, and that the more modest figure of that platform is close to what URL-only detection achieves on realistic data.

Two problems separate benchmark accuracy from scan-time performance. First, many benchmark features are not available at scan time: they describe the rendered page, its links and forms, or third-party reputation services, all of which require contacting the destination, which is precisely what a pre-interaction check must avoid. Second, benchmarks can contain collection artifacts that let a model separate the classes without learning anything about phishing, a form of shortcut learning [6] and one of the pitfalls identified for machine learning in security [7]. We find both problems in widely used benchmarks. On the UCI Phishing Websites dataset [8], accuracy falls from 97.2% to 73.7% when only features computable from the URL string are used. In PhiUSIIL [9], every one of the 134,850 legitimate URLs is an `https://www.` homepage, so the one-line rule "phishing unless the URL starts with `https://www.`" reaches 99.1% accuracy, within one percentage point of the best reported models. The same rule exceeds 99.6% on two further public datasets. A model trained on PhiUSIIL performs at chance level on four other datasets and flags virtually every legitimate URL that contains a path, the form in which QR codes for menus, payments, and forms typically appear. For QR/NFC security, such a detector would block legitimate codes while offering little real protection.

Building on this analysis, we propose **StackShield**, a scan-time phishing URL detector trained and evaluated on PhreshPhish [10], the only benchmark in our study without a trivial shortcut. StackShield uses stacked generalization: a character n-gram model produces an out-of-fold phishing score, which an XGBoost classifier combines with 116 explainable features describing URL structure, domain and hosting, and brand impersonation. On a forward-in-time test set of 166,542 URLs collected after the training period, StackShield raises balanced accuracy from 89.1% to 96.8% (AUC 99.5%) and detects 94.4% of phishing URLs while blocking only 1% of legitimate ones, compared with 71.9% for the baseline feature set. It classifies a URL in 25 ms and explains each decision with SHAP [11].

This paper makes the following contributions:

- **A scan-time audit of phishing URL benchmarks.** We separate benchmark features by when they become available (URL string, network lookups, rendered page) and quantify how much reported accuracy depends on information unavailable before a user opens a page. We document the construction artifacts of PhiUSIIL, including its homepage-only legitimate class and two inconsistent feature-extraction conventions.
- **A cross-dataset study on five public datasets.** Using a validated re-implementation of PhiUSIIL's URL features, we train on each dataset and test on every other, with leakage controls. We show that a simple string rule solves three of the five datasets, that models do not transfer between them, and that PhiUSIIL-trained models misclassify the URL forms typical of QR codes.
- **StackShield, an explainable scan-time detector.** We design leakage-safe feature groups, including brand and typosquat features covering global and Gulf-region brands, and stack them with a character n-gram model. We evaluate on data collected after the training period and report operating points relevant to deployment, such as recall at 1% and 0.1% false-positive rates.
- **Evidence on what does and does not help.** Ablation with confidence intervals and McNemar tests measures each feature group, including the lookalike, brand-impersonation, and typosquat cases where the baseline fails. A heterogeneous stacking ensemble of XGBoost, Random Forest, and a neural network adds no practical gain at about 35 times the training cost.
- **Open problems for QR/NFC deployment.** We identify cases that current benchmarks do not cover but QR/NFC deployments encounter routinely, such as legitimate pages on free hosting platforms and legitimate shortened or dynamic-QR links.
- **Open artifacts.** We will release the feature extractor, including its validation against PhiUSIIL's published feature values, the scripts for the benchmark audit and cross-dataset study, and the StackShield training and evaluation pipeline upon acceptance, to support reproduction and the evaluation of future QR/NFC phishing detectors.

The remainder of this paper is organized as follows. Section 2 reviews related work on QR/NFC threats, phishing URL detection, and dataset bias in security machine learning. Section 3 defines the threat model and the information available at scan time. Section 4 presents the benchmark audit, and Section 5 the cross-dataset study. Section 6 describes StackShield, and Section 7 reports its evaluation. Section 8 discusses open problems for QR/NFC deployment and the limitations of this work, and Section 9 concludes.

## 2 Related Work

This work draws on four lines of research: security threats specific to QR codes and NFC, machine learning for phishing URL detection, ensemble learning and explainability, and the quality and evaluation of datasets used in security machine learning. The first three establish the need for scan-time URL classification and the techniques

available for it; the fourth explains why reported accuracy on public benchmarks can overstate what such classifiers achieve in practice.

## 2.1 QR Code and NFC Security Threats

QR codes and NFC tags do not let users verify the destination or payload before interaction, which makes them attractive vectors for phishing, malware delivery, and other social engineering attacks [12, 13]. Kieseberg et al. [14] showed that the visual appearance of a QR code gives no indication of its content, leaving users and automated systems exposed to malicious code substitution. Krombholz et al. [12] proposed one of the first taxonomies of QR-code attacks, covering URL redirection, code replacement, and social engineering.

User studies confirm that these attacks work. Vidas et al. [3] found that about 85% of participants followed URLs from publicly posted QR codes without verification. Sharevski et al. [4] reported that 67% of participants who scanned a malicious QR code disclosed personal information on a counterfeit COVID-19 vaccination portal, and Geisler and Pöhn [15] observed that susceptibility to quishing remains high in a real-world study. Beyond phishing pages, Han et al. [16] showed with the MEDUSA attack that in-app QR scanners can be abused to reach sensitive functionality and commit financial fraud.

NFC faces comparable problems at the protocol and application layers. Alexiou et al. [17] analyzed NFC relay attacks with probabilistic model checking and proposed distance-bounding countermeasures; Roland et al. [18] demonstrated a practical relay attack against Google Wallet; and Miller [19] showed that Android devices could perform malicious actions after reading crafted NFC tags. Onumadu and Abroshan [13] survey current NFC threats in payment systems, including relay, cloning, eavesdropping, and data manipulation.

Across these studies, the central weakness is the same: users cannot verify the destination before interacting. Most work characterizes attacks or targets user awareness; comparatively little addresses automatic classification of the decoded URL before it is opened, which is the setting of this paper.

## 2.2 Machine Learning for Phishing URL Detection

Machine learning detectors can recognize previously unseen phishing sites, which blocklists cannot. They differ mainly in what information they use: the URL string (lexical and structural features), network and registration data (DNS, WHOIS, certificates, reputation), and the content of the rendered page (HTML, forms, links, visual appearance). Ma et al. [20] showed early that lexical and host-based URL features support accurate classification. Later work combined many feature types: Mohammad et al. [8] used 30 engineered features, many of them page- or reputation-based, and PhiUSIIL [9] combines URL, domain, and HTML features with a similarity index and incremental learning. Detectors that use only the URL string include classical feature-based approaches [21, 22] and character-level models that learn directly from the URL text, such as URLNet [23]. Brand impersonation through lookalike and typosquatted domains is a long-studied phishing technique [24], but it is rarely modeled explicitly in URL classifiers.

The distinction between these information sources matters for QR/NFC security. A pre-interaction check must decide before the destination is contacted, so page-content features, and in a strict setting also network lookups, are unavailable. Published results rarely separate accuracy by the information each feature requires, which makes it difficult to judge how well a detector would work at scan time.

Work specific to QR and NFC has taken other directions. Trad and Chehab [25] classify QR codes from image characteristics without decoding them, reaching 91% accuracy, and Sarkhi and Mishra [26] propose a lightweight deep learning model for detecting QR-code attacks. In the NFC domain, Vishwakarma et al [27] and Tagle [28] detect fraudulent contactless payment transactions from transaction-level features.

Al-Mannai et al. [5] built a browser-based pre-interaction platform that classifies URLs extracted from QR codes and NFC tags, with XGBoost reaching 90.15% accuracy on a balanced proprietary dataset of 20,000 URLs. That study used a single dataset and a single classifier, and did not examine how its accuracy compared with, or would transfer to, public benchmarks.

## 2.3 Ensemble Learning and Explainability

Ensembles are widely used in phishing detection. Bagging (e.g., Random Forest) reduces variance, boosting (e.g., XGBoost) corrects errors sequentially, and stacked generalization [29] trains a meta-learner on the out-of-fold predictions of base models. Stacking is usually applied to heterogeneous classifiers trained on the same features. It can also combine models that see different representations of the input, for example a character-level text model and a feature-based model; StackShield uses stacking in this second sense.

Explainability is important in security applications, where analysts and users need to understand why a URL was blocked. SHAP [11] attributes each prediction to input features using Shapley values and is widely used both to explain models and to rank features.

Several phishing studies report that stacking ensembles and SHAP-based feature selection improve accuracy on a given benchmark [30–32]. As the next subsection discusses, such improvements are hard to interpret when the benchmark itself can be solved by simple shortcuts, because small accuracy differences near the ceiling may reflect the dataset rather than the method.

## 2.4 Dataset Quality, Shortcut Learning, and Evaluation Pitfalls

Models can reach high accuracy by exploiting *shortcuts*: features that separate the classes in a dataset for reasons unrelated to the task and that fail to generalize [6]. Arp et al. [7] identify sampling bias, spurious correlations, data snooping, and inappropriate baselines among the most common pitfalls in machine learning for security, and Pendlebury et al. [33] show that ignoring the temporal order of samples inflates results in malware classification. For phishing, Dalton et al. [10] report that existing public datasets suffer from label noise, leakage between training and test data, and unrealistic base rates, and introduce PhreshPhish with benchmarks designed to reduce these problems.

Two gaps remain. First, the widely used URL benchmarks have not been audited for scan-time conditions: to our knowledge, no study quantifies how much of their reported accuracy depends on features unavailable before a page is opened, or documents their construction artifacts. Related observations have appeared in dataset documentation but not in the peer-reviewed literature: the authors of the X-PhiDe dataset report that HTTPS presence is highly discriminative in PhiUSIIL but much less so in another dataset [34].

Second, although detectors are usually trained and tested on splits of the same dataset, recent work has begun to show that they do not transfer. Mia et al. [35] trained XGBoost on 20 features shared by two public phishing datasets and found that accuracy fell to 51–59% when the model was trained on one dataset and tested on the other; SHAP analysis showed that the same feature could push predictions toward phishing in one dataset and toward legitimate in the other. Rashid et al. [36] observed that URL classifiers work well only when training and test sets come from the same dataset, and proposed an unsupervised domain adaptation framework that aligns features between a source and a target dataset to improve transfer. These studies establish that the problem exists and offer a way to adapt models to new data, but they do not explain why models fail to transfer, separate features by when they become available, or relate the failures to the URL forms that a deployed detector encounters. This paper addresses these questions across five datasets, and evaluates its detector on data collected after the training period.

Table 1 compares representative studies along the dimensions that matter for scan-time detection: the input they use, whether they are restricted to information available before a page is opened, whether they are evaluated across datasets or over time, and whether their predictions are explained. Studies on QR and NFC security either classify QR images or payment transactions, or, like the platform of Al-Mannai et al. [5], rely on a single proprietary dataset. General phishing detectors such as those evaluated on UCI and PhiUSIIL use page-content or reputation features and are evaluated within a single dataset. PhreshPhish introduces a temporal evaluation but not a scan-time restriction. None of the studies combines a scan-time feature restriction, evaluation across several public datasets and over time, and explainable predictions in the QR/NFC setting.

To address these gaps, this paper first audits widely used phishing URL benchmarks under scan-time conditions and evaluates detectors across five public datasets, and then proposes StackShield, a detector restricted to information available at scan time. StackShield stacks the out-of-fold score of a character n-gram model with explainable features describing URL structure, domain and hosting, and brand impersonation, and combines them in an XGBoost classifier whose decisions are explained with SHAP. Unlike previous studies, it is trained and evaluated on data free of the shortcuts identified in the audit, tested on URLs collected after the training period, compared against a heterogeneous stacking ensemble of XGBoost, Random Forest, and a Multi-Layer Perceptron, and assessed on the URL forms that QR/NFC deployments encounter, such as lookalike domains, brand impersonation, and legitimate pages with paths.

**Table 1** Comparison of representative studies on QR/NFC security and phishing URL detection.

| Study | Application | Input | Dataset(s) | Scan-time only | Cross-dataset / temporal eval. | Explainability | Main contribution |
|---|---|---|---|---|---|---|---|
| [27] | NFC payments | Transactions | Proprietary | – | No | No | ML for NFC payment fraud detection |
| [8] | Phishing websites | URL, page, reputation features | UCI (30 features) | No | No | No | Self-structuring neural network |
| [22] | Malicious URLs | URL features | Proprietary | Partly | No | No | Feature-based malicious URL classification |
| [9] | Phishing URLs | URL, domain, HTML features | PhiUSIIL | No | No | No | Similarity index and incremental learning |
| [28] | NFC payments | Transactions | Proprietary | – | No | No | Real-time NFC fraud detection |
| [26] | QR security | QR images | Public | Yes | No | No | Lightweight deep learning for QR attack detection |
| [37] | QR security | Literature | – | – | – | – | Review of QR phishing detection methods |
| [25] | QR security | QR images | Public | Yes | No | No | Quishing detection without decoding |
| [10] | Phishing websites | URL and HTML | PhreshPhish | No | Temporal | No | High-quality dataset and realistic benchmarks |
| [5] | QR/NFC | URL features | Proprietary | Partly | No | No | Browser-based XGBoost detection platform |
| **StackShield (this work)** | **QR/NFC** | **URL string** | **5 public datasets** | **Yes** | **Both** | **Yes (SHAP)** | **Scan-time benchmark audit and stacked explainable detector** |

# 3 Threat Model and Scan-Time Information

A pre-interaction phishing check is only useful if it can decide before the user reaches the destination, and this constraint determines which information a detector may use. This section defines the adversary and the deployment setting considered in this work, organizes the information that phishing detectors commonly rely on into three tiers according to when it becomes available, and formulates scan-time phishing detection as a classification problem with deployment-oriented objectives. These definitions underlie the benchmark audit in Section 4, the cross-dataset study in Section 5, and the design of StackShield in Section 6.

## 3.1 Adversary and Deployment Setting

We consider an adversary who creates and distributes malicious QR codes or NFC tags in physical and digital environments, including public spaces, payment terminals, parking meters, restaurant tables, product labels, printed advertisements, and emails. The adversary may place stickers over legitimate codes, rewrite NFC tags, or distribute new codes in social engineering campaigns. The goal is to lead victims to phishing pages, credential-harvesting forms, or malware downloads.

The adversary controls the URL embedded in the code or tag and the infrastructure behind it. In particular, the adversary can register new or lookalike domains, including typosquatted variants of brand names; obtain valid TLS certificates at no cost, so that phishing URLs commonly begin with `https://` and, if desired, `https://www.`; host pages on free site-builder and cloud platforms whose domains are shared with legitimate users; and hide the destination behind URL shorteners or dynamic-QR redirection services. We do not assume that the adversary knows the detection model.

The user scans or taps with a standard mobile device and does not inspect the decoded URL. As in the platform of Al-Mannai et al. [5], the detector operates as a *pre-interaction screening layer*: after the URL is decoded and before the browser opens it, the URL is classified, and the user is warned or blocked if it is predicted to be phishing. Scanning, URL extraction, and the user interface are unchanged; this work concerns only the classification component.

Attacks that do not rely on a malicious destination URL are out of scope. These include NFC relay and cloning attacks [17, 18], vulnerabilities in scanner applications themselves [16], and non-URL payloads such as Wi-Fi configurations or contact cards. Adversarial manipulation of the classifier by an adversary who knows the model is left to future work.

## 3.2 Information Tiers

Phishing detectors use information from three sources, which become available at different points in the interaction (Table 2).

- **Tier 1: URL string.** Everything computable from the decoded URL text: its length, characters, tokens, scheme, host, subdomains, top-level domain, path, and query. It is available immediately after decoding and requires no network access.

**Table 2** Information tiers used by phishing detectors and their availability at scan time.

| Tier | Examples | Available |
|---|---|---|
| 1: URL string | Length, character counts, tokens, scheme, subdomains, TLD, path, query | Immediately after decoding; no network access |
| 2: Network lookups | DNS, registration age, TLS certificate, reputation, redirects, `robots.txt`, favicon | After contacting third parties or the destination server |
| 3: Rendered page | HTML, forms, links, scripts, iframes, page title | Only after loading attacker-controlled content |

- **Tier 2: Network lookups.** Information obtained by querying external services or the destination server without rendering the page: DNS records, domain registration data, TLS certificate properties, reputation and traffic rankings, redirect chains, and the presence of files such as `robots.txt` or a favicon. Some of these queries contact the adversary's infrastructure directly, and all of them add latency.
- **Tier 3: Rendered page.** The HTML, scripts, forms, links, and visual appearance of the destination page. Obtaining it requires loading attacker-controlled content, which is what a pre-interaction check is meant to prevent, and phishing kits can serve different content to automated crawlers than to victims.

StackShield uses Tier 1 information only. This is the strictest setting, but it has three advantages for QR/NFC deployment: the decision is made without any connection to the destination, it adds no network latency, and it does not reveal to the adversary that a URL is being checked. In Section 4 we use the same tiers to measure how much of the accuracy reported on public benchmarks depends on Tier 2 and Tier 3 information.

## 3.3 Problem Formulation

Let $u$ be a URL decoded from a QR code or NFC tag, and let $y \in \{0, 1\}$ be its label, where $y = 1$ denotes phishing and $y = 0$ legitimate. A feature map $\varphi$ computes a vector $\mathbf{x} = \varphi(u) \in \mathbb{R}^d$ from Tier 1 information only. A classifier

$$f : \mathbb{R}^d \to [0, 1] \tag{1}$$

estimates the probability that $u$ is phishing, and $u$ is flagged if $f(\mathbf{x}) \geq \tau$ for a decision threshold $\tau$.

Two properties of the deployment shape how $f$ is evaluated. First, the proportion of phishing among scanned URLs is unknown and differs between datasets, so we report balanced accuracy, the mean of the true positive rate (TPR) and the true negative rate, and the area under the ROC curve (AUC) alongside accuracy. Second, false alarms are costly: users who are frequently warned about legitimate codes learn to ignore warnings. We therefore also report the TPR achieved when the false positive

rate (FPR) is limited to a small value $\alpha$,

$$\mathrm{TPR@}\alpha = \mathrm{TPR}(\tau_\alpha), \qquad \tau_\alpha = \min\{\tau : \mathrm{FPR}(\tau) \le \alpha\}, \tag{2}$$

with $\alpha = 1\%$ and $\alpha = 0.1\%$.

Finally, a detector must work on URLs it has not seen and on phishing campaigns that appear after it was trained. We therefore evaluate models on datasets other than the one they were trained on (Section 5), with URLs occurring in the training data removed from each test set, and on test data collected after the training period (Section 7).

# 4 Auditing Phishing URL Benchmarks under Scan-Time Conditions

Before designing a scan-time detector, we examine how well accuracy reported on public benchmarks reflects the setting of Section 3. We re-evaluate the two benchmarks used most often in URL-based phishing detection, the UCI Phishing Websites dataset [8] and PhiUSIIL [9], with a protocol designed to avoid leakage, and ask two questions: how much of their accuracy depends on Tier 2 and Tier 3 information, and whether their construction allows a model to separate the classes without learning anything about phishing.

## 4.1 Re-evaluation Protocol

**Labels.** In both datasets the label value 1 denotes a *legitimate* URL. We remap the labels so that 1 denotes phishing throughout, consistent with Eq. (1). PhiUSIIL then contains 100,945 phishing (42.8%) and 134,850 legitimate URLs; UCI contains 4,898 phishing (44.3%) and 6,157 legitimate instances.

**Feature tiers.** We assign every feature to a tier of Table 2. In UCI, 8 of the 30 features are computable from the URL string (IP address in host, URL length, shortening service, `@` symbol, double-slash redirection, prefix-suffix hyphen, subdomain count, and `https` token in the domain); 11 require network lookups (TLS certificate state, registration length, abnormal URL, open ports, domain age, DNS record, web traffic, page rank, search-engine index, inbound links, and blocklist reports); and 11 require the rendered page (favicon, request URL, anchor URLs, links in tags, server form handler, submission to e-mail, redirects, mouse-over and right-click scripts, pop-ups, and iframes). We evaluate three nested feature sets: U1 (Tier 1, 8 features), U2 (Tiers 1–2, 19), and U3 (all 30). PhiUSIIL provides 50 numeric features. We exclude 24 that describe page content (Tier 3) and, as a precaution, four that are computed from statistics of the dataset's own URL collection and can therefore encode label information (URL similarity index, TLD legitimacy probability, URL character probability, and character continuation rate). The remaining features form two sets: T1 (18 features computable from the URL string) and T2 (T1 plus favicon, `robots.txt`, and two redirect counts, which require fetching resources from the destination).

**Models and training.** We train nine classifiers (XGBoost, Random Forest, gradient boosting, decision tree, MLP, RBF-kernel SVM, $k$-nearest neighbours, logistic

**Table 3** Re-evaluation of two benchmarks by information tier (accuracy, mean over five seeds). "Consistent difference" counts the seeds in which McNemar's test finds the stacking ensemble significantly different from the best single classifier ($p < 0.05$).

| Dataset | Feature set (tiers) | #Feat. | Best single | Stacking | Consistent diff. |
|---|---|---|---|---|---|
| UCI | U3 (1–3) | 30 | 97.2% (RF) | 97.2% | 0/5 |
| UCI | U2 (1–2) | 19 | 94.7% (RF) | 94.7% | 0/5 |
| UCI | U1 (1) | 8 | 73.7% (XGB, RF) | 73.7% | 0/5 |
| PhiUSIIL | T2 (1–2) | 22 | 99.75% (XGB) | 99.77% | 1/5 |
| PhiUSIIL | T1 (1) | 18 | 99.77% (XGB) | 99.78% | 0/5 |

regression, and Gaussian naive Bayes), with feature standardization inside the pipeline for the scale-sensitive models; the SVM is trained on a stratified subsample of 20,000 URLs because of its quadratic cost. We also train a heterogeneous stacking ensemble with XGBoost, Random Forest, and MLP as base learners and logistic regression as meta-learner on five-fold out-of-fold probabilities, both on all features and on a SHAP-ranked subset. The subset size is selected by five-fold cross-validation *within the training split*, using log-loss and the one-standard-error rule, so the test set is never used for model selection.

**Evaluation.** Each experiment is repeated over five stratified 80/20 splits (seeds 0–4). We report means over seeds, phishing-class precision and recall, and compare classifiers with McNemar's test on each seed, counting a difference as consistent only if it is significant ($p < 0.05$) in a majority of seeds.

## 4.2 How Much Accuracy Depends on Information Unavailable at Scan Time

Table 3 summarizes the results. On UCI, the best single classifier (Random Forest) reaches 97.2% accuracy with all 30 features, in line with published results (e.g., 97.64% with SHAP-based feature selection and a meta-ensemble [32]). Restricting the model to Tiers 1–2 reduces accuracy to 94.7%, and restricting it to the URL string reduces it to 73.7%. Most of UCI's discriminative information therefore lies in page content and third-party reputation data that a pre-interaction check cannot use; its URL-string features are coarse, three-valued indicators that separate the classes poorly.

PhiUSIIL behaves in the opposite way. With the 18 URL-string features alone, XGBoost reaches 99.77% accuracy, and adding the four fetch-based features does not improve it. Even logistic regression, a linear model, reaches 99.6%. A benchmark on which URL-string features alone give near-perfect linear separability warrants closer inspection, which we carry out in Section 4.3. The stacking ensemble is not consistently better than the best single classifier on any feature set. On PhiUSIIL it is significantly

**Table 4** URL-string properties of the two classes in PhiUSIIL (percentage of URLs in each class).

| Property | Legitimate | Phishing |
|---|---|---|
| Starts with `https://www.` | 100.0 | 2.4 |
| Uses `https://` | 100.0 | 48.7 |
| Bare homepage (no path, no query) | 100.0 | 71.8 |
| Same URL-to-domain length difference as the template | 100.0 | 11.1 |
| Features computed after dropping the last character | 100.0 | 51.8 |

better than Random Forest in all five seeds, but by only 0.05–0.07 percentage points, and it is statistically indistinguishable from XGBoost. SHAP-based selection reduces the PhiUSIIL feature set from 18 to about 8 features without a significant loss, but on UCI the cross-validated selection retains all features of U1 and U2 and 20–25 of the 30 features of U3. In short, on these benchmarks the choice of feature tier changes accuracy by up to 23 percentage points, while the choice between a strong single model and a stacking ensemble changes it by less than 0.1.

## 4.3 Construction Artifacts in PhiUSIIL

Table 4 compares the two classes of PhiUSIIL on simple properties of the URL string. Every one of the 134,850 legitimate URLs begins with `https://www.` and is a bare homepage, with no path or query. Only 2.4% of phishing URLs begin with `https://www.`, and 51.3% use plain `http://`. A further consequence is that, for every legitimate URL, the difference between the URL length and the domain length has the same value, which holds for only 11.1% of phishing URLs. The legitimate class thus consists of URLs built from a single template, `https://www.` followed by a domain name.

This template is enough to solve the benchmark. Table 5 lists the accuracy of increasingly simple classifiers. The one-line rule "phishing unless the URL starts with `https://www.`" reaches 98.96% accuracy on the full dataset (99.10% after the deduplication used in Section 5), and logistic regression on three features (URL length, domain length, and HTTPS) reaches 98.05%. A threshold on URL length alone reaches only 74.0%, so the shortcut is not simply that phishing URLs are longer: it is the combination of scheme, prefix, and length that identifies the legitimate template. The best machine learning models add less than one percentage point to a string comparison.

The SHAP analysis of the T1 model confirms that the model has learned this template. In Fig. 1, each dot is one training URL, its horizontal position is the SHAP value of the feature for that URL, and its color is the feature value. The four most important features are domain length, HTTPS, URL length, and the letter ratio, and each of them encodes part of the template. URLs without HTTPS receive large positive SHAP values, pushing them toward phishing, whereas HTTPS alone shifts a URL only slightly toward legitimate. Long URLs are pushed toward phishing, as any path or query departs from the bare-homepage form. Short domain names are also

**Table 5** Accuracy of simple classifiers on PhiUSIIL (five-fold cross-validation unless noted).

| Classifier | Accuracy |
|---|---|
| Single threshold on URL length | 74.0% |
| Logistic regression on URL length, domain length, HTTPS | 98.05% |
| Rule: phishing unless URL starts with `https://www.` (no training) | 98.96% |
| XGBoost, 18 URL-string features (T1; held-out test, five seeds) | 99.77% |

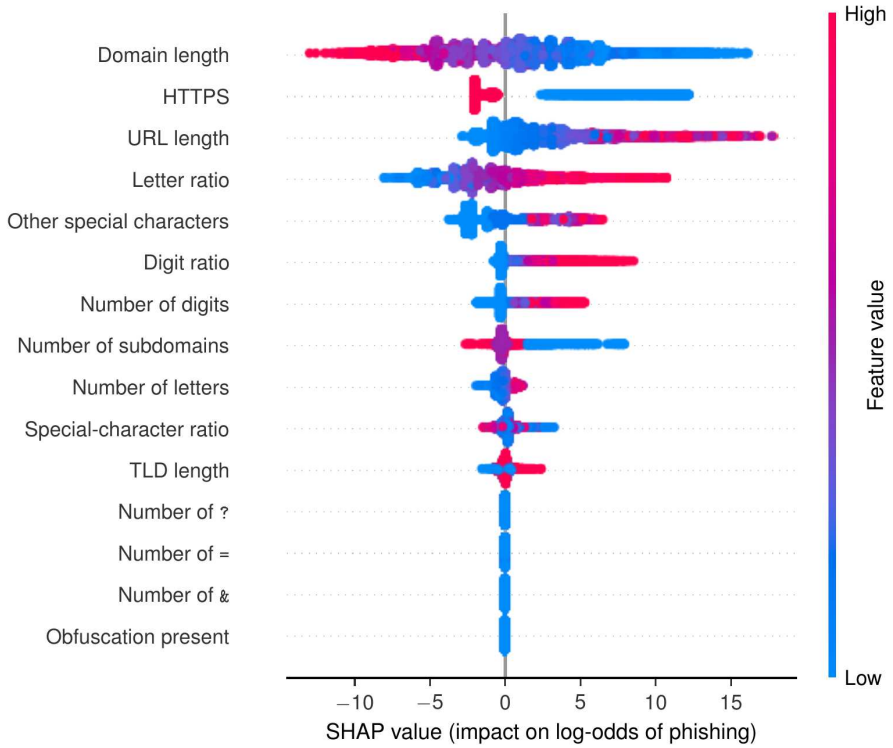



**Fig. 1** SHAP summary of XGBoost trained on the 18 URL-string features of PhiUSIIL (first seed). Positive values push the prediction toward phishing. The top-ranked features are those that distinguish the `https://www.`-plus-domain template of the legitimate class.

pushed toward phishing, because the domain of every legitimate URL includes the `www.` prefix of the template. None of the top features describes what the domain name says, such as a brand name or a suspicious word; the model separates the classes by URL form alone.

The dataset contains a second, smaller inconsistency, shown in the last row of Table 4. To compute URL features for other datasets (Section 5), we re-implemented PhiUSIIL's 18 URL-string features and validated them against the published values.

**Table 6** Phishing detection rate (recall) on PhiUSIIL by URL group, 18 URL-string features, mean over five seeds. Groups A–C overlap; group D is the complement of A.

| Phishing group | $n$ per test split | LR | XGBoost | Stacking |
|---|---|---|---|---|
| A: starts with `https://www.` | 484 | 64.3% | 79.1% | 81.0% |
| B: same length pattern as the template | 2,241 | 95.2% | 96.3% | 96.3% |
| C: bare homepage | 14,495 | 98.8% | 99.3% | 99.4% |
| D: all other phishing | 19,705 | 99.96% | 99.98% | 99.99% |

The re-implementation reproduces the values of all legitimate URLs exactly, but only when the last character of each URL is removed before the features are computed. The same convention holds for 52,299 phishing URLs, whereas the remaining 48,644 phishing URLs were processed using the full URL. The two classes, therefore, passed partly through different extraction procedures. Training XGBoost on features recomputed with one consistent convention gives almost the same result (99.76% instead of 99.77% accuracy in five-fold cross-validation), so this inconsistency is not what makes the benchmark easy, but it is further evidence that the classes were assembled separately.

## 4.4 Where Models Trained on PhiUSIIL Fail

If a model has learned the template rather than phishing, its errors should concentrate on phishing URLs that resemble the template. Table 6 confirms this. For phishing URLs that start with `https://www.`, the stacking ensemble detects 81.0%, XGBoost 79.1%, and logistic regression 64.3%; for all other phishing URLs, detection exceeds 99.9% for all three models. About 92 of the roughly 100 phishing URLs missed per test split fall in the first group, although it contains only 1.0% of the test URLs. Inspection of the missed URLs shows that many would be recognizable to a human: typosquats and brand impersonations such as `steamcomnununity[.]com`, `dhlrewardscards[.]com`, and `supportcopyrght-ig[.]net`, and domains on abused TLDs such as `mygov3-au[.]top`. The model misses them because its features count characters rather than describe what the domain name says.

This failure matters for QR/NFC security. Adversaries can obtain a domain with a `www` subdomain and a valid TLS certificate at no cost (Section 3), so the URLs that PhiUSIIL-trained models trust most are among the easiest for an adversary to produce. Conversely, legitimate QR codes for menus, payments, and forms usually point to pages with paths, a form that is absent from PhiUSIIL's legitimate class. Section 5 tests the consequences of both observations on independent data.

The audit yields three findings. First, on UCI the accuracy reported for full-feature models depends mostly on information unavailable before a page is opened: URL-string features alone give 73.7%. Second, PhiUSIIL can be solved almost completely by a single string rule, because its legitimate class follows one template, so accuracy near 99.8% on this benchmark says little about phishing detection. Third, on both

**Table 7** Datasets used in the cross-dataset study (after cleaning).

| Dataset | URLs | Phishing | Split |
|---|---|---|---|
| PhiUSIIL [9] | 231,747 | 41.8% | Stratified 80/20 |
| PhreshPhish [10] | 652,974 | 43.7% | Official, forward in time |
| KaggleURLs [38] | 617,123 | 48.8% | Stratified 80/20 |
| Mendeley [39] | 437,725 | 21.0% | Stratified 80/20 |
| Extra111k [40] | 114,283 | 14.2% | Stratified 80/20 |

benchmarks the choice of classifier, including stacking and SHAP-based selection, has a negligible effect compared with the choice of information. These findings motivate the cross-dataset study in the next section and the design of StackShield, which relies on realistic training data and on features that describe the content of the domain name rather than only its length.

# 5 Cross-Dataset Study

The audit in Section 4 shows that one benchmark depends on information unavailable at scan time and another can be solved by a string rule. This section asks whether these are isolated cases and what they mean for a deployed detector. We compare five public datasets on the same scan-time representation, test for shortcuts in each, and train on every dataset and test on every other. A detector that has learned phishing, rather than the collection procedure of its training data, should retain much of its accuracy on independent data.

## 5.1 Datasets and Common Representation

We use five public datasets that provide raw URLs (Table 7). PhiUSIIL [9] is described in Section 4. PhreshPhish [10] was built specifically to reduce label noise, leakage, and unrealistic base rates; we use its official split, in which the test URLs (September–December 2025) were collected after the training URLs (July 2024–September 2025). The Benign and Malicious URLs dataset [38] ("KaggleURLs") is a balanced collection; its malicious class may include URL types other than phishing, so we refer to it as malicious rather than phishing.

its malicious class may include URL types other than phishing, so we refer to it as malicious rather than phishing. The Mendeley Phishing URL dataset [39] combines phishing feeds with top-site lists. The Phishing URL Detection dataset [40] ("Extra111k") does not document its sources, so we use it as supplementary evidence only.

All datasets are reduced to the raw URL and a binary label, with 1 denoting phishing. URLs are normalized (lower case, without scheme, leading `www.`, and trailing slash); duplicates and URLs that occur with both labels are removed before splitting. For each dataset we compute the 18 URL-string features of PhiUSIIL with our re-implementation (Section 4.3), using one consistent extraction convention for all URLs.

**Table 8** URL shape by dataset and class (percentage of URLs, after deduplication) and accuracy of the rule "phishing unless the URL starts with `https://www.`". Leg. = legitimate, Ph. = phishing or malicious.

| | **HTTPS** | | `https://www.` | | **Bare homepage** | | **Rule** | |
|---|---|---|---|---|---|---|---|---|
| **Dataset** | Leg. | Ph. | Leg. | Ph. | Leg. | Ph. | Acc. | Bal. acc. |
| PhiUSIIL | 100.0 | 48.5 | 100.0 | 2.2 | 100.0 | 72.0 | 99.1 | 98.9 |
| KaggleURLs | 100.0 | 6.1 | 99.8 | 0.5 | 12.8 | 10.5 | 99.7 | 99.7 |
| Mendeley | 100.0 | 6.5 | 99.8 | 0.7 | 13.8 | 8.2 | 99.7 | 99.5 |
| PhreshPhish | 97.8 | 87.8 | 56.9 | 5.5 | 7.4 | 45.1 | 73.3 | 75.7 |
| Extra111k | 34.8 | 91.7 | 24.5 | 2.3 | 76.0 | 42.1 | 34.9 | 61.1 |

We train XGBoost on each dataset's training split and evaluate it on the test split of every dataset; for PhiUSIIL we additionally train logistic regression and the heterogeneous stacking ensemble of Section 4. For every train–test pair, test URLs whose normalized form occurs in the training set are removed. Because the proportion of phishing differs strongly between datasets, we report balanced accuracy and AUC, together with the phishing recall and the false positive rate (FPR).

## 5.2 Every Dataset Carries a Collection Signature

Table 8 compares the URL shape of the two classes in each dataset and the accuracy of the one-line rule "phishing unless the URL starts with `https://www.`". The rule reaches at least 99.1% on three of the five datasets. In KaggleURLs and Mendeley, 99.8% of legitimate URLs start with `https://www.`, while only about 6% of malicious or phishing URLs use HTTPS at all. The two datasets also share their source of legitimate URLs: when one is used for training and the other for testing, about 65,000 overlapping URLs have to be removed from the test set.

The two remaining datasets have their own, different signatures. In Extra111k, only 34.8% of legitimate URLs use HTTPS while 91.7% of phishing URLs do, which reverses the pattern of the other datasets. PhreshPhish has no single shortcut of this kind: most of its phishing URLs use HTTPS (87.8%), as current phishing does, and its legitimate URLs are varied, with only 7.4% bare homepages. Its classes nevertheless differ in shape as well; in particular, bare homepages are common among its phishing URLs (45.1%), and its legitimate URLs are longer (median 55 characters) than its phishing URLs (median 39).

## 5.3 Models Do Not Transfer Between Datasets

Fig. 2 shows the balanced accuracy (a) and AUC (b) of XGBoost for all 25 train–test pairs. Each row corresponds to a training dataset and each column to a test dataset, so the outlined diagonal gives in-domain performance and every other cell measures transfer. Blue cells indicate performance above chance, white cells performance close to chance (0.5), and red cells performance worse than chance, where a model systematically ranks legitimate URLs as more suspicious than phishing URLs. Every model performs well on the test split of its own dataset (balanced accuracy 88.3–99.8%), but

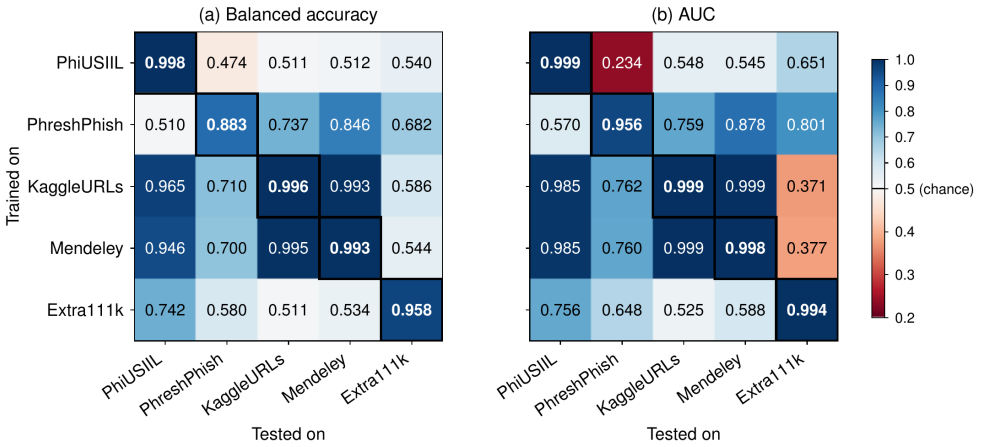



**Fig. 2** Cross-dataset performance of XGBoost with 18 URL-string features: (a) balanced accuracy and (b) AUC. Rows give the training dataset, columns the test dataset; outlined cells on the diagonal are in-domain results. The color scale is centered at 0.5, the performance of random guessing; red cells indicate performance worse than chance. Test URLs that occur in the training set are removed for every pair.

performance off the diagonal drops sharply. The only pairs that transfer well are from KaggleURLs and Mendeley to each other and to PhiUSIIL (balanced accuracy 94.6–99.5%), the three datasets whose legitimate URLs start with `https://www.`; models trained on KaggleURLs or Mendeley reach only 70.0–71.0% on PhreshPhish. Transfer within this group therefore reflects a shared shortcut rather than shared knowledge about phishing.

The PhiUSIIL-trained model fails on every other dataset (Table 9). Its balanced accuracy is 47.4–54.0%, the level of random guessing, because it labels almost everything as phishing: 91.8–99.3% of legitimate URLs are flagged. On PhreshPhish its AUC is 0.234, well below the 0.5 of a random classifier, meaning that it ranks legitimate URLs as *more* suspicious than phishing URLs. This follows directly from the artifacts in Section 4.3: PhiUSIIL teaches that anything longer than a bare `https://www.` domain is suspicious, whereas in PhreshPhish legitimate URLs are the longer ones. Neither a linear model nor the stacking ensemble avoids this failure, which confirms that the problem lies in the training data rather than in the choice of classifier. Logistic regression transfers slightly better than XGBoost (48.6–59.0% balanced accuracy), because it fits the template less precisely, but it remains close to chance on every external dataset. Similar drops have been reported for other pairs of phishing datasets [35].

The failure is not specific to PhiUSIIL. The model trained on PhreshPhish, the dataset with the weakest shortcut, transfers best to the other datasets (68.2–84.6% balanced accuracy on KaggleURLs, Mendeley, and Extra111k), but reaches only 51.0% on PhiUSIIL: it learned from its own data that bare homepages are more often phishing, and every legitimate URL in PhiUSIIL is a bare homepage. The models trained on KaggleURLs and Mendeley, in turn, have AUC below 0.4 on Extra111k, whose

**Table 9** Models trained on PhiUSIIL, tested on the other datasets: balanced accuracy for three classifiers, and phishing recall and FPR for XGBoost.

| | Balanced accuracy | | | XGBoost | |
|---|---|---|---|---|---|
| **Test dataset** | **LR** | **XGBoost** | **Stacking** | **Recall** | **FPR** |
| PhiUSIIL (in-domain) | 99.6% | 99.8% | 99.8% | 99.7% | 0.0% |
| PhreshPhish | 48.6% | 47.4% | 47.2% | 94.0% | 99.3% |
| KaggleURLs | 55.8% | 51.1% | 50.5% | 100.0% | 97.9% |
| Mendeley | 56.2% | 51.2% | 50.6% | 100.0% | 97.5% |
| Extra111k | 59.0% | 54.0% | 53.6% | 99.8% | 91.8% |

**Table 10** Share of URLs flagged as phishing by URL form: FPR for legitimate groups, recall for phishing groups (percent). "PU" denotes XGBoost trained on PhiUSIIL; "ID" (in-domain) denotes XGBoost trained on the test dataset's own training split.

| | **PhreshPhish** | | **KaggleURLs** | | **Mendeley** | | **Extra111k** | |
|---|---|---|---|---|---|---|---|---|
| **Group** | PU | ID | PU | ID | PU | ID | PU | ID |
| Legit. with path (FPR) | 100.0 | 3.1 | 100.0 | 0.1 | 100.0 | 0.1 | 100.0 | 2.3 |
| Legit. homepage (FPR) | 91.7 | 23.4 | 82.8 | 0.6 | 81.5 | 0.2 | 89.1 | 0.8 |
| Phish. `https://www.` (recall) | 49.7 | 67.2 | 98.3 | 15.9 | 98.0 | 11.0 | 91.6 | 65.5 |
| Phish. other (recall) | 100.0 | 83.4 | 100.0 | 99.8 | 100.0 | 99.5 | 100.0 | 93.4 |

HTTPS pattern is the reverse of theirs. Each dataset teaches its own collection choices, in whichever direction they point.

## 5.4 Consequences for QR/NFC Deployment

Table 10 breaks down the behavior of the PhiUSIIL-trained model by the URL forms that matter for QR/NFC deployments, and compares it with a model trained on each test dataset itself. The PhiUSIIL-trained model flags every legitimate URL that contains a path, on all four datasets, and 81.5–91.7% of legitimate homepages. QR codes for restaurant menus, payments, parking, and forms almost always encode URLs with paths, so such a detector would block essentially every legitimate code a user scans. At the same time, it detects only 49.7% of the PhreshPhish phishing URLs that start with `https://www.`.

The in-domain models reveal the same weakness from the other side. On KaggleURLs and Mendeley, where almost no phishing URL starts with `https://www.`, the in-domain models detect only 11.0–15.9% of the few that do. Even on PhreshPhish, the in-domain model detects only 67.2% of such URLs and flags 23.4% of legitimate homepages.

Finally, none of the datasets represents legitimate shortened links. Their test splits together contain 804 phishing URLs on known URL-shortening or dynamic-QR services, but only five legitimate ones. Any model trained on these data therefore learns

that a shortened link indicates phishing, although many legitimate QR codes, especially dynamic codes whose destination can be changed after printing, use exactly such services.

Three of five public datasets can be solved by a one-line string rule, and each dataset carries a collection signature that models learn instead of phishing. As a result, a model with 99.8% accuracy on its own benchmark can perform at chance level on another and would block virtually every legitimate QR code. PhreshPhish has the weakest shortcut, contains predominantly HTTPS phishing as observed today, and provides a forward-in-time split; it is the most suitable of the five for developing a scan-time detector. We therefore train and evaluate StackShield on PhreshPhish (Sections 6 and 7), with particular attention to the groups where the models above fail: phishing URLs that start with `https://www.`, legitimate homepages, and legitimate URLs with paths. The cross-dataset gaps also show that no single public dataset is sufficient to validate a QR/NFC detector, a point we return to in Section 8.

# 6 StackShield

StackShield is a scan-time phishing URL detector that uses Tier 1 information only (Section 3). Its design follows from the failures observed in Sections 4 and 5. First, it is trained on PhreshPhish, the dataset with the weakest collection shortcut and a forward-in-time split. Second, because models based on character counts missed lookalike and brand-impersonating URLs, StackShield adds features that describe what the URL says: its structure, its domain and hosting, and its relation to known brands. Third, because such hand-crafted features cannot anticipate every suspicious pattern, StackShield stacks them with a character n-gram model that learns substrings directly from the data. This section describes the data preparation, the five feature groups, the stacking architecture, and the leakage controls. Fig. 3 gives an overview of the architecture.

## 6.1 Data Preparation

We use the PhreshPhish dataset with its official split. URLs are cleaned as in Section 5: normalized URLs that occur more than once are kept once, URLs that occur with both labels are removed, and a URL that appears in both official splits is kept in the training split only, so that no test URL occurs in the training data. The resulting training set contains 486,432 URLs collected between 2 July 2024 and 8 September 2025, and the test set contains 166,542 URLs collected between 8 September and 16 December 2025. All development decisions, including feature design and hyperparameters, use the training split only; the test split is used once, for the evaluation in Section 7.

## 6.2 Feature Groups

StackShield computes 117 features in five groups (Table 11). All of them are derived from the URL string; none requires network access.

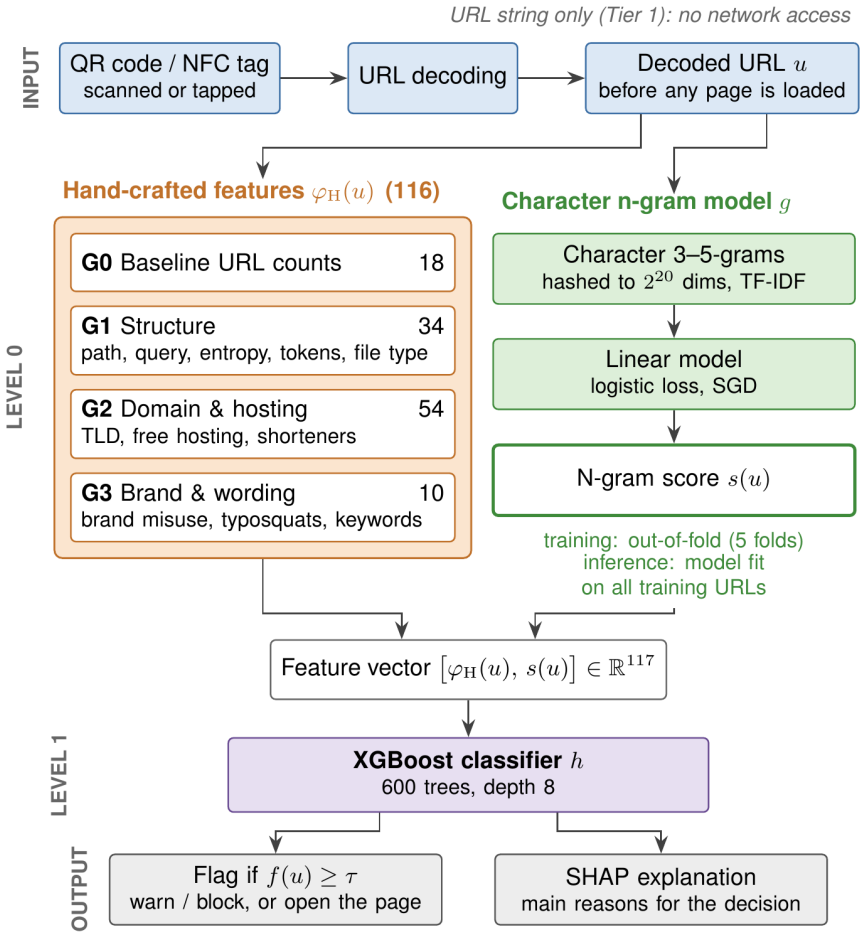



**Fig. 3** Architecture of StackShield. The decoded URL is processed without network access. At Level 0, 116 hand-crafted features (G0–G3) are computed, and a character n-gram model produces a phishing score $s(u)$, which is computed out of fold for training URLs. At Level 1, an XGBoost classifier combines both into the phishing probability $f(u)$; the URL is flagged if $f(u) \geq \tau$, and SHAP values explain the decision.

**G0: baseline.** The 18 URL-string features of PhiUSIIL, computed with the validated re-implementation of Section 4.3. They serve as the reference point for all improvements.

**G1: structure.** These features describe the shape of the URL beyond character counts. The URL is parsed into host, path, query, and fragment, and the host is split into subdomain, registered domain, and public suffix using the Public Suffix List. Randomness in generated domain names is captured by the Shannon entropy

$$H(s) = -\sum_{c} p_c \log_2 p_c, \tag{3}$$

**Table 11** Feature groups of StackShield. All features are computed from the URL string (Tier 1).

| Group | # | Features |
|---|---|---|
| G0 Baseline | 18 | PhiUSIIL URL-string features: URL and domain length, IP host, TLD length, subdomain count, obfuscation, counts and ratios of letters, digits, `=`, `?`, `&`, and other special characters, HTTPS |
| G1 Structure | 34 | Path and query length, path depth, number of parameters, fragment, hyphens and digits in the host, Shannon entropy of host and URL, number and length of tokens, upper-case characters, `@`, explicit port, `//` in path, punycode, lengths of registered domain and subdomain, number of subdomain labels, `www` subdomain, missing scheme, file extension (12 indicators) |
| G2 Domain and hosting | 54 | Top-level domain (the 50 most frequent in the training split plus "other"), multi-label public suffix, free hosting or site-builder platform, URL-shortening or dynamic-QR service |
| G3 Brand and wording | 10 | Brand in URL, brand as registered domain, brand used outside the registered domain, brand in subdomain, brand in path, brand embedded in a longer domain name, minimum edit distance to a brand (absolute and relative), typosquatting indicator, number of sensitive words |
| G4 Character n-grams | 1 | Out-of-fold phishing probability of a linear model on character 3–5-grams |

where $p_c$ is the relative frequency of character $c$ in the string $s$, computed for the host and for the full URL. Tokens are the maximal alphanumeric substrings of the URL.

**G2: domain and hosting.** The top-level domain is one-hot encoded over the 50 most frequent values in the training split; the vocabulary is determined by frequency alone, without using labels. Two binary features indicate whether the registered domain belongs to one of 62 free hosting, site-builder, cloud-storage, and form services (e.g., `pages.dev`, `weeblysite.com`, `webflow.io`, `docs.google.com`) or to one of 35 URL-shortening and dynamic-QR services. These platforms let adversaries publish pages under domains shared with legitimate users.

**G3: brand and wording.** These features target impersonation, the main cause of the errors observed in Section 4. We compiled a list of 94 frequently impersonated brands, covering technology and social media, payment and banking, delivery, cryptocurrency, telecommunications, and government services, including brands from the Gulf region such as QNB, Ooredoo, Qatar Airways, and Absher. For a URL with registered domain label $\ell$ (e.g., `paypal` in `paypal.com`), the features indicate whether a brand name appears in the URL, whether $\ell$ is itself a brand, and whether a brand appears in the subdomain, the path, or embedded in a longer $\ell$ (e.g., `dhlrewardscards`). Typosquatting is captured by the edit distance to the closest

brand among the 79 brand names of at least five characters,

$$d(\ell) = \min_{b} \mathrm{Lev}(\ell, b), \qquad r(\ell) = \frac{d(\ell)}{|b^*|}, \tag{4}$$

where Lev is the Levenshtein distance and $b^*$ the closest brand. A URL is marked as a typosquat if $0 < d(\ell)$ and $r(\ell) \leq 0.25$; for example, `steamcomnununity` has distance 3 to `steamcommunity` ($r = 0.21$). Finally, we count occurrences of 37 words common in phishing URLs, such as *login*, *verify*, *wallet*, and *airdrop*.

**G4: character n-grams.** Hand-crafted features cannot anticipate every suspicious pattern. We therefore represent each URL by its character 3- to 5-grams, hashed into $2^{20}$ dimensions and weighted by sublinear TF-IDF (with document frequencies from the training split), and train a linear classifier with logistic loss by stochastic gradient descent (regularization $\alpha = 2 \times 10^{-6}$, 15 epochs). This model learns substrings associated with phishing, such as fragments of lookalike brand names, without manual specification. Its output enters StackShield as a single feature, as described next.

## 6.3 Stacked Architecture

Fig. 3 shows how the components fit together. A URL decoded from a QR code or NFC tag is processed without any network access. At Level 0, two branches process it in parallel: the left branch computes the 116 hand-crafted features of groups G0–G3, and the right branch passes the URL's character n-grams to a linear model that outputs a single phishing score $s(u)$. At Level 1, the two branches are joined into one feature vector, and an XGBoost classifier produces the phishing probability $f(u)$. The URL is flagged if $f(u)$ exceeds the threshold $\tau$, and SHAP values provide the reasons for the decision. The only difference between training and inference lies in how $s(u)$ is computed, as described next.

StackShield combines the n-gram model and the hand-crafted features by stacked generalization [29]. Let $\varphi_{\mathrm{H}}(u) \in \mathbb{R}^{116}$ denote the hand-crafted features of groups G0–G3, and let $g$ denote the n-gram model. For training URLs, the n-gram score $s(u)$ is obtained out of fold: the training split is divided into five stratified folds, and $s(u)$ is predicted by a model $g^{(-k)}$ trained on the four folds that do not contain $u$. For test URLs, $s(u)$ is predicted by a model $g$ trained on the full training split. The final classifier is

$$f(u) = h\big([\varphi_{\mathrm{H}}(u),\ s(u)]\big), \tag{5}$$

where $h$ is an XGBoost classifier with 600 trees, learning rate 0.08, maximum depth 8, row subsampling 0.9, and column subsampling 0.8. Out-of-fold scores are essential: if $s(u)$ were computed by a model that had seen $u$, the n-gram score of training URLs would be overconfident, and $h$ would learn to rely on it more than is justified on new URLs.

This architecture differs from the heterogeneous stacking ensemble evaluated in Section 4, in which several classifiers are trained on the *same* features and a meta-learner combines their outputs. In StackShield, the two levels see *different representations* of the URL, raw character sequences and interpretable features, and the second level can learn when to trust the n-gram score, for instance for URLs on shared

**Algorithm 1** StackShield training and inference

**Require:** Training URLs $\{(u_i, y_i)\}_{i=1}^{N}$ collected before time $t_0$
**Ensure:** Trained n-gram model $g$ and classifier $h$
Normalize URLs; remove duplicates and URLs with conflicting labels
Compute hand-crafted features $\varphi_{\mathrm{H}}(u_i)$ (G0–G3); fix the TLD vocabulary from the training URLs
Fit the hashing and TF-IDF transformation of character 3–5-grams on the training URLs
Split the training URLs into five stratified folds
**for** $k = 1, \ldots, 5$ **do**
    Train n-gram model $g^{(-k)}$ on all folds except $k$
    $s(u_i) \leftarrow g^{(-k)}(u_i)$ for all $u_i$ in fold $k$ ▷ out-of-fold score
**end for**
Train $g$ on all training URLs
Train XGBoost $h$ on $\left[\varphi_{\mathrm{H}}(u_i), s(u_i)\right]$ with labels $y_i$
**Inference** for a decoded URL $u$:
    $\mathbf{x} \leftarrow \left[\varphi_{\mathrm{H}}(u),\, g(u)\right]$; **return** phishing if $h(\mathbf{x}) \geq \tau$

hosting platforms or with brand names in unusual positions. For comparison, Section 7 also evaluates the heterogeneous ensemble (XGBoost, Random Forest with 200 trees, and an MLP with hidden layers of 128 and 64 units, combined by logistic regression) on the same 117 features.

Algorithm 1 summarizes training and inference.

## 6.4 Explanations

For each prediction, StackShield reports the SHAP values [11] of its 117 inputs, computed exactly for the tree ensemble with TreeSHAP. A positive value indicates that a feature pushed the prediction toward phishing. Because 116 inputs are interpretable features, explanations can name concrete reasons, such as a brand name outside the registered domain, a free hosting platform, or an unusually long random subdomain. The n-gram score appears as a single input, so its contribution indicates how much of a decision rests on learned character patterns. For global analysis, the importance of feature $j$ is the mean absolute SHAP value over a set of $N$ URLs,

$$I_j = \frac{1}{N} \sum_{i=1}^{N} |\phi_{ij}| \,, \tag{6}$$

where $\phi_{ij}$ is the SHAP value of feature $j$ for URL $i$.

## 6.5 Leakage Controls

Because Sections 4 and 5 show how easily evaluation can be distorted, we summarize the measures that keep the test split independent: (i) duplicates are removed on normalized URLs, and no test URL occurs in the training split; (ii) the test URLs

were collected after all training URLs; (iii) the TF-IDF weights and the TLD vocabulary are derived from the training split only, and all hyperparameters were fixed in advance, without using the test split; (iv) the n-gram score of training URLs is computed out of fold; and (v) the brand, sensitive-word, hosting, and shortener lists were compiled from public knowledge of phishing techniques, not from the test data.

### 6.6 Baseline and Model Variants

To measure the contribution of each part, we train XGBoost with the same hyperparameters on cumulative feature sets: G0 (the baseline), G0–G1, G0–G2, G0–G3, and G0–G4 (StackShield), as well as on StackShield's features with one group removed at a time. We also evaluate the n-gram model alone and the heterogeneous stacking ensemble. The baseline G0 corresponds to the PhreshPhish in-domain model of Section 5, but with the hyperparameters above (more and deeper trees than the library defaults); this raises its balanced accuracy from 88.3% to 89.1% and makes it the stronger reference.

## 7 Evaluation

This section evaluates StackShield on the PhreshPhish test split, which contains 166,542 URLs collected after all training URLs (Section 6). We report overall performance with confidence intervals, the contribution of each feature group, a comparison of stacking designs, performance on the URL groups where earlier models failed, stability over time, explanations, transfer to the other datasets, and computational cost. Unless stated otherwise, a URL is flagged as phishing if $f(u) \geq 0.5$. Confidence intervals are 95% percentile intervals from 200 bootstrap resamples of the test split, and paired comparisons use McNemar's test.

### 7.1 Overall Performance

Table 12 reports all configurations. The baseline, XGBoost on the 18 PhiUSIIL URL features (G0), reaches 89.1% balanced accuracy and an AUC of 95.8%. This is close to the 90.15% accuracy reported by Al-Mannai et al. [5], which was obtained on a different dataset, and it is a realistic reference for URL-only detection with count-based features. StackShield raises balanced accuracy to 96.8% and AUC to 99.5%, reduces the false positive rate from 4.7% to 1.2%, and increases phishing recall from 83.0% to 94.8%. The confidence intervals of the two models are far apart, and McNemar's test gives $p < 10^{-300}$: StackShield corrects 13,542 of the baseline's errors and introduces 1,420 new ones.

For deployment, the operating points at low false positive rates matter more than accuracy at a fixed threshold, because users who receive frequent false warnings learn to ignore them. If at most 1% of legitimate URLs may be flagged, StackShield detects 94.4% of phishing URLs, compared with 71.9% for the baseline. At 0.1%, one false alarm per thousand legitimate scans, it detects 85.3%, twice the baseline's 42.4%.

**Table 12** Results on the PhreshPhish test split (166,542 URLs, September–December 2025). All models except the n-gram model alone and the heterogeneous ensemble use XGBoost with identical hyperparameters. R@1% and R@0.1% denote phishing recall at a false positive rate of 1% and 0.1% (Eq. 2). Intervals are 95% bootstrap confidence intervals.

| **Configuration** | **#Feat.** | **Bal. acc.** | **95% CI** | **AUC** | **Recall** | **FPR** | **Prec.** | **R@1%** | **R@0.1%** |
|---|---|---|---|---|---|---|---|---|---|
| G0 (baseline) | 18 | 89.13 | [89.00, 89.28] | 95.79 | 82.98 | 4.72 | 93.55 | 71.89 | 42.37 |
| G0–G1 | 52 | 91.64 | [91.50, 91.76] | 97.28 | 86.41 | 3.12 | 95.81 | 80.27 | 67.26 |
| G0–G2 | 106 | 93.70 | [93.59, 93.82] | 98.01 | 89.43 | 2.02 | 97.34 | 86.99 | 74.25 |
| G0–G3 | 116 | 94.06 | [93.93, 94.17] | 98.35 | 89.99 | 1.88 | 97.54 | 87.94 | 75.95 |
| **StackShield (G0–G4)** | **117** | **96.81** | **[96.73, 96.89]** | **99.48** | **94.83** | **1.22** | **98.47** | **94.42** | **85.27** |
| StackShield without G1 | 83 | 96.61 | – | 99.38 | 94.62 | 1.39 | 98.25 | 93.62 | 80.90 |
| StackShield without G2 | 63 | 96.77 | – | 99.46 | 94.81 | 1.27 | 98.40 | 94.25 | 84.26 |
| StackShield without G3 | 107 | 96.77 | – | 99.46 | 94.78 | 1.25 | 98.43 | 94.20 | 84.37 |
| StackShield without G4 | 116 | 94.06 | [93.93, 94.17] | 98.35 | 89.99 | 1.88 | 97.54 | 87.94 | 75.95 |
| N-gram model alone | – | 94.86 | [94.76, 94.97] | 99.04 | 91.42 | 1.69 | 97.81 | 88.34 | 54.18 |
| Heterogeneous ensemble | 117 | 96.86 | [96.78, 96.95] | 99.47 | 94.94 | 1.23 | 98.46 | 94.38 | 73.15 |

**Table 13** Paired comparisons on the test split (McNemar's test). "A only" counts URLs classified correctly by A but not by B, and vice versa.

| **A** | **B** | **A only** | **B only** | $p$ |
|---|---|---|---|---|
| G0–G1 | G0 | 6,163 | 2,123 | $< 10^{-300}$ |
| G0–G2 | G0–G1 | 4,767 | 1,485 | $< 10^{-300}$ |
| G0–G3 | G0–G2 | 1,466 | 915 | $1.8 \times 10^{-29}$ |
| StackShield | G0–G3 | 5,376 | 1,127 | $< 10^{-300}$ |
| StackShield | N-gram model alone | 4,048 | 1,043 | $< 10^{-300}$ |
| StackShield | G0 | 13,542 | 1,420 | $< 10^{-300}$ |
| Heterogeneous ensemble | StackShield | 431 | 358 | 0.010 |

## 7.2 Contribution of the Feature Groups

Each cumulative step in Table 12 improves balanced accuracy significantly. Table 13 reports, for each pair of models, the number of test URLs that only one of the two classifies correctly. In every step, the richer model corrects between 1.6 and 4.8 times as many URLs as it newly misclassifies, and all differences are significant ($p < 10^{-28}$). Measured by balanced accuracy, structure features add 2.5 percentage points, domain and hosting features 2.1, brand and wording features 0.4, and the n-gram score 2.8. The brand features contribute least overall, but they target a specific error type, whose detection they improve substantially (Section 7.4).

Removing one group at a time from StackShield gives a different picture. Without the n-gram score, balanced accuracy falls by 2.75 percentage points; without any single hand-crafted group, it falls by at most 0.20 points, and without G2 or G3 by only 0.04 points, within the confidence interval of the full model. The n-gram model therefore captures much of the information in each hand-crafted group, and each group is partly redundant given the others. The groups are not redundant at strict operating points, however: removing G1 lowers recall at 0.1% FPR from 85.3% to 80.9%, and removing G2 or G3 lowers it by about one point. The hand-crafted features make StackShield reliable where false alarms must be rare, and they are what make its decisions explainable (Section 7.6).

## 7.3 Stacking Designs

The comparison between the n-gram model, the hand-crafted features, and their combination shows why StackShield stacks them. The n-gram model alone (94.9%) is better than all hand-crafted groups together (94.1%), yet StackShield improves on both (96.8%), and the difference is largest at strict operating points: at 0.1% FPR the n-gram model alone detects only 54.2% of phishing, the hand-crafted features 76.0%, and StackShield 85.3%. The two representations thus make different errors, and the second-level classifier learns to combine them.

The heterogeneous stacking ensemble of XGBoost, Random Forest, and an MLP on the same 117 features reaches 96.86% balanced accuracy, 0.05 percentage points

**Table 14** Share of URLs flagged as phishing on the PhreshPhish test split, by URL group: false positive rate for legitimate groups and recall for phishing groups (percent).

| Group | $n$ | G0 | G0–G3 | StackShield | Het. ens. |
|---|---|---|---|---|---|
| *Legitimate (FPR, lower is better)* | | | | | |
| Homepage | 8,064 | 22.33 | 9.76 | 5.89 | 5.89 |
| With path or query | 83,163 | 3.01 | 1.11 | 0.76 | 0.78 |
| On hosting platform | 199 | 28.14 | 41.71 | 33.67 | 33.67 |
| *Phishing (recall, higher is better)* | | | | | |
| Starts with `https://www.` | 8,944 | 66.26 | 73.62 | 85.11 | 85.02 |
| Other | 66,371 | 85.23 | 92.19 | 96.14 | 96.28 |
| On hosting platform | 13,018 | 96.51 | 99.72 | 99.59 | 99.65 |
| Brand outside registered domain | 11,343 | 71.31 | 88.46 | 96.99 | 97.12 |
| Typosquatted domain | 108 | 64.81 | 96.30 | 94.44 | 96.30 |
| Shortened link | 187 | 91.44 | 100.00 | 99.47 | 100.00 |

above StackShield. The difference is statistically significant ($p = 0.010$) but practically negligible, as the two models disagree on only 789 of 166,542 URLs. The ensemble took 70 minutes to train, compared with about 2 minutes for StackShield's XGBoost stage, and its recall at 0.1% FPR is 12 points lower (73.2% versus 85.3%). We therefore use XGBoost as StackShield's second level.

## 7.4 Performance on Critical URL Groups

Table 14 reports performance on the URL groups identified as problematic in Sections 4 and 5. StackShield improves detection of phishing URLs that start with `https://www.` from 66.3% to 85.1%, the group that exploits the shortcut of earlier benchmarks. It detects 97.0% of phishing URLs that use a brand name outside the registered domain, compared with 71.3% for the baseline, and 94.4% of typosquatted domains, compared with 64.8%. Much of this gain is achieved without the n-gram score (G0–G3 reaches 88.5% and 96.3% on these two groups), and the n-gram score adds further gains on brand misuse. On typosquats, StackShield (94.4%) is slightly below G0–G3 (96.3%); with 108 URLs this is a difference of two URLs.

On the legitimate side, StackShield flags only 0.76% of legitimate URLs with a path or query, the form typical of QR codes for menus, payments, and forms, compared with 3.0% for the baseline and 100% for the PhiUSIIL-trained model of Section 5. False alarms on legitimate homepages fall from 22.3% to 5.9%.

One group remains problematic. Of the 199 legitimate URLs hosted on free hosting or site-builder platforms, StackShield flags 33.7%, more than the baseline (28.1%), because the hosting features learn that such platforms are strongly associated with phishing: 13,018 phishing URLs in the test split are hosted on them. Small businesses frequently publish menus and forms on exactly these platforms, so this weakness is directly relevant to QR/NFC deployment; we discuss it in Section 8. The same caution applies to shortened links: StackShield detects all but one phishing URL on shortening services, but PhreshPhish contains almost no legitimate shortened links, so its false positive rate on them cannot be measured.

**Table 15** Balanced accuracy by month of collection on the PhreshPhish test split (percent).

| Month | URLs | Phishing | G0 | StackShield |
|---|---|---|---|---|
| September 2025 | 31,626 | 13.4% | 76.10 | 92.73 |
| October 2025 | 113,043 | 46.3% | 89.98 | 97.14 |
| November 2025 | 20,860 | 89.9% | 89.59 | 97.03 |

## 7.5 Stability over Time

Table 15 reports balanced accuracy by month of collection. StackShield remains between 92.7% and 97.1% over the three months following the end of the training period, with no downward trend. September 2025, the first test month, has the lowest value for both models; it also has an unusually low share of phishing (13.4%), so the difference should not be interpreted as drift. The remaining 1,013 test URLs, from December 2025, are not reported separately because the per-month analysis requires at least 500 URLs of both classes.

## 7.6 Explanations

Fig. 4 shows the global SHAP importance of StackShield on 20,000 test URLs, aggregated by feature group in panel (a) and for the 20 most important individual features in panel (b). The n-gram score accounts for 48.9% of the total mean absolute SHAP value, followed by the structure features (27.8%), the baseline features (15.0%), the domain and hosting features (4.8%), and the brand and wording features (3.5%). After the n-gram score, the most important features are path length, the number of question marks, the letter ratio, subdomain length, path depth, and the length of the registered domain. The relative edit distance to a brand ranks 14th overall, although it applies to a small fraction of URLs. Unlike the PhiUSIIL-trained model (Fig. 1), StackShield does not depend on the presence of `https://www.`, and the `www` indicator ranks only 18th.

## 7.7 Transfer to Other Datasets

Table 16 evaluates StackShield and the baseline, both trained on PhreshPhish, on the test splits of the other four datasets, with overlapping URLs removed. StackShield improves on the baseline on every dataset, by 2.9–8.5 points of balanced accuracy on KaggleURLs, Mendeley, and Extra111k and by 6.8–26.0 points of AUC overall. On PhiUSIIL, AUC rises from 0.596 to 0.856, so StackShield ranks PhiUSIIL's URLs much better, but at the default threshold it still flags 87.7% of PhiUSIIL's legitimate URLs. As Section 5 showed, PhreshPhish associates bare homepages with phishing, while every legitimate URL in PhiUSIIL is a bare homepage. Better features reduce, but do not remove, the effect of differences between data collections.

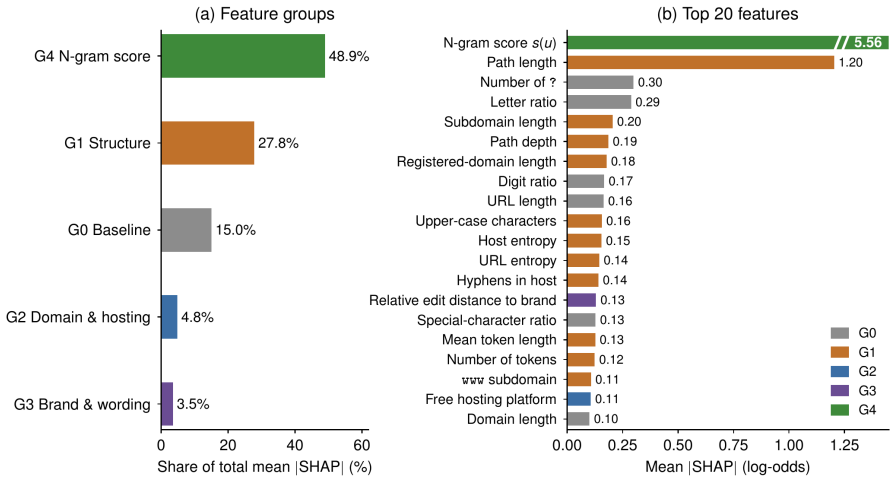



**Fig. 4** Global SHAP importance of StackShield, computed as the mean absolute SHAP value (Eq. 6) on 20,000 URLs from the PhreshPhish test split: (a) share of the total importance per feature group; (b) the 20 most important features, colored by group. The bar of the n-gram score is truncated.

**Table 16** Models trained on PhreshPhish, evaluated on the other datasets (percent). SS = StackShield.

| | | **AUC** | | **Bal. acc.** | | **Recall** | | **FPR** | |
|---|---|---|---|---|---|---|---|---|---|
| **Test dataset** | $n$ | G0 | SS | G0 | SS | G0 | SS | G0 | SS |
| PhiUSIIL | 46,077 | 59.60 | 85.57 | 48.70 | 53.98 | 89.95 | 95.66 | 92.54 | 87.69 |
| KaggleURLs | 123,355 | 74.07 | 80.88 | 73.03 | 75.96 | 63.30 | 65.64 | 17.24 | 13.72 |
| Mendeley | 87,467 | 87.32 | 94.36 | 83.85 | 88.34 | 85.13 | 90.47 | 17.43 | 13.79 |
| Extra111k | 21,935 | 80.97 | 93.16 | 70.52 | 79.02 | 86.67 | 94.95 | 45.64 | 36.91 |

## 7.8 Computational Cost

Table 17 summarizes the cost of StackShield. Classifying a single URL, from the raw string to the decision, including feature extraction, the n-gram score, and XGBoost inference, takes a median of 25.4 ms (95th percentile 57.9 ms) in unoptimized Python on a standard cloud CPU (Google Colab). This is well below the delay a user would notice after scanning and well below the time needed to load a web page. The trained models occupy about 11 MB. These measurements correspond to server-side classification, as in [5]; on-device performance was not measured. Feature extraction and training are one-time costs, incurred when the model is built or retrained; only the inference time applies to each scan.

These results leave two weaknesses, legitimate pages on free hosting platforms and incomplete transfer to differently collected datasets, which we discuss in Section 8.

**Table 17** Computational cost of StackShield (Google Colab CPU).

| Step | Cost |
|---|---|
| Hand-crafted features, 652,974 URLs | 4.9 min |
| N-gram score (out of fold, 5 folds) | 2.5 min |
| XGBoost training (StackShield) | 115 s |
| Heterogeneous ensemble training (for comparison) | 69.8 min |
| Inference per URL, median / 95th percentile | 25.4 / 57.9 ms |
| Model size (XGBoost / n-gram weights) | 6.7 / 4.2 MB |

# 8 Discussion

## 8.1 What Benchmark Accuracy Measures

The results of Sections 4–7 show that accuracy on a phishing URL benchmark depends at least as much on the benchmark as on the detector. On UCI, the information tier changes accuracy by 23 percentage points, whereas the choice of classifier changes it by less than 0.1. On PhiUSIIL, a string comparison comes within one point of the best models, and three of the five datasets we examined can be solved by the same rule. A model can therefore reach 99.8% on one benchmark and chance level on another, and improvements of a few tenths of a percentage point on a single benchmark, including those previously reported for ensembles and feature selection, may reflect a benchmark's collection procedure rather than better phishing detection.

These findings suggest simple practices for evaluating phishing URL detectors: (i) state which information tier each feature requires, and report results for the tier available in the intended deployment; (ii) report trivial baselines, such as a scheme or prefix rule, alongside machine learning models; (iii) remove duplicates on normalized URLs and exclude overlapping URLs between training and test data; (iv) evaluate on at least one dataset collected independently of the training data, and preferably on data collected after the training period; and (v) report recall at low false positive rates and performance on critical URL groups, not only accuracy at a fixed threshold. Each of these practices changed our own conclusions. Our code, which implements all of them, will be released upon acceptance.

The results also place the platform of Al-Mannai et al. [5] in context. Its 90.15% accuracy with URL features appeared modest next to benchmark results of 97–99%, but it is close to what count-based URL features achieve on realistic data (89.1% balanced accuracy on PhreshPhish). StackShield raises this to 96.8% under the same scan-time restriction, and the gain comes from what the features describe (structure, hosting, brands, and learned character patterns) rather than from a more complex classifier.

## 8.2 Implications for QR/NFC Deployment

For a pre-interaction screening layer, the relevant question is how much phishing is detected at a false alarm rate that users will tolerate. At a threshold corresponding to 0.1% FPR on the test data, StackShield would warn on about one in a thousand legitimate scans while detecting 85.3% of phishing; at 1%, it would detect 94.4%. The appropriate threshold depends on the setting: a payment application may accept more warnings than a restaurant menu scanner. Because thresholds estimated on one dataset shift on another (Section 7), a deployed system should calibrate its threshold on URLs from its own environment.

StackShield's explanations can also support the user interface. Instead of an unexplained block, the platform can report the main reasons for a warning, for example that the URL uses a well-known brand name outside that brand's domain, or that the page is hosted on a free site-builder platform. Such explanations help users judge borderline cases and make false alarms easier to accept.

## 8.3 Open Problems

Two URL categories are central to QR/NFC deployments but poorly covered by current data. The first is legitimate pages on free hosting and site-builder platforms. StackShield flags a third of the 199 such pages in the PhreshPhish test split, because the same platforms host 13,018 of its phishing URLs. Small restaurants, shops, and event organizers often publish menus and forms on exactly these platforms, so the true false positive rate for QR codes in such settings may be considerably higher than the 1.2% measured overall.

The second is legitimate shortened and dynamic-QR links. Dynamic QR codes, whose destination can be changed after printing, route every scan through a redirection service, and many legitimate codes are created this way. Across the five datasets, the test splits contain 804 phishing URLs on shortening or dynamic-QR services but only five legitimate ones. Any model trained on these data learns that such links indicate phishing, and none of the datasets can measure how often legitimate dynamic codes would be blocked. Resolving the redirect before classification would reveal the final destination, but it requires contacting a third-party service (Tier 2) and, if the chain continues, the adversary's server. Resolving only known redirection services through their official interfaces is a possible compromise that should be evaluated.

Both problems call for a dataset collected from the QR/NFC channel itself: legitimate codes found in public places, businesses, and government services, and malicious codes from quishing reports, with the decoded URLs and, where safe, their resolved destinations. Such a dataset would also allow thresholds to be calibrated for the base rate of phishing actually encountered by scanning users.

Finally, StackShield has not been evaluated against an adversary who adapts URLs to the model. Some features, such as brand edit distance and hosting platform, are costly for an adversary to avoid, whereas others, such as path length or character ratios, are easy to change. Measuring robustness to such adaptation, and retraining regularly as campaigns change, are necessary steps before deployment.

### 8.4 Limitations

Our study has several limitations. First, none of the datasets was collected from QR codes or NFC tags; we evaluate URL forms typical of the QR/NFC channel, not URLs actually decoded from codes in the field. Second, the labels of all five datasets are subject to noise, the malicious class of KaggleURLs may include URL types other than phishing, and the sources of Extra111k are undocumented. Third, StackShield is trained and tested on a single dataset, and its test period covers about three months; its feature groups and hyperparameters were not tuned on other datasets, which is why we report cross-dataset results separately. Fourth, the results in Section 7 come from a single training run with confidence intervals obtained by bootstrapping the test split; they do not capture variation due to random initialization, although the effects we report are large relative to the intervals. Fifth, the brand, hosting, and shortener lists were compiled manually and will need maintenance; a brand missing from the list cannot contribute to the brand features, although the n-gram score may still capture it. Sixth, latency was measured on a cloud CPU in unoptimized Python, corresponding to server-side classification, and not on a mobile device.

## 9 Conclusion

QR codes and NFC tags conceal their destination until after the user interacts, so protection depends on classifying the URL at scan time. This paper showed that accuracy reported on public phishing URL benchmarks is a poor guide to how well such a classifier works. On the UCI Phishing Websites dataset, most of the reported accuracy depends on information unavailable before a page is opened: URL-string features alone give 73.7% instead of 97.2%. In PhiUSIIL, every legitimate URL follows a single `https://www.` template, so a one-line rule reaches 99.1% accuracy, and the same rule solves two further public datasets. Across five datasets, models do not transfer: a model with 99.8% accuracy on PhiUSIIL performs at chance level on the other four and flags every legitimate URL with a path, the form in which most legitimate QR codes appear.

Building on this analysis, we proposed StackShield, a scan-time detector that stacks the out-of-fold score of a character n-gram model with 116 explainable features describing URL structure, domain and hosting, and brand impersonation. On 166,542 PhreshPhish URLs collected after its training period, it raises balanced accuracy from 89.1% to 96.8% and detects 94.4% of phishing at a 1% false positive rate and 85.3% at 0.1%. It corrects most failures on lookalike `https://www.` URLs, brand impersonation, and typosquatting, flags fewer than 1% of legitimate URLs with paths, and explains each decision with SHAP in about 25 ms. A heterogeneous stacking ensemble adds no practical benefit at 35 times the training cost: the gains come from what the model sees, not from how many models are combined.

Future work will address the open problems identified in this paper: collecting a dataset of URLs decoded from QR codes and NFC tags in real environments, including legitimate dynamic codes and small-business pages on free hosting platforms; evaluating safe resolution of redirection services; integrating StackShield into a QR/NFC

screening platform and measuring its performance on mobile devices; assessing robustness against adaptive adversaries and strategies for continual retraining; and studying with users how explained warnings affect their decisions.

**Use of generative AI.** ChatGPT (OpenAI) was used to improve the language and clarity of the manuscript and to assist with literature search. All references were verified by the authors against their original sources. The authors reviewed and edited all AI-assisted text and take full responsibility for the content of this article. No AI tool was used to generate research data, results, or conclusions.

## Declarations


- **Funding:** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.
- **Competing interests:** The authors have no competing interests to declare that are relevant to the content of this article.
- **Ethics approval and consent to participate:** Not applicable.
- **Consent for publication:** Not applicable.
- **Data availability:** All datasets used in this study are publicly available from their original sources: PhiUSIIL [9], PhreshPhish [10], Benign and Malicious URLs [38], Mendeley Phishing URL [39], and Phishing URL Detection (111K) [40]. Raw phishing URLs are not redistributed.
- **Materials availability:** Not applicable.
- **Code availability:** The feature extractor, benchmark audit, cross-dataset evaluation, and StackShield pipeline will be made publicly available upon acceptance; the link has been withheld to preserve anonymity during review.


## References


[1] ISO/IEC: Information technology – automatic identification and data capture techniques – QR Code bar code symbology specification. Standard ISO/IEC 18004:2015, International Organization for Standardization, Geneva, Switzerland (2015)

[2] ISO/IEC: Cards and security devices for personal identification – contactless proximity objects – part 1: Physical characteristics. Standard ISO/IEC 14443-1:2018, International Organization for Standardization (2018). https://www.iso.org/standard/73596.html

[3] Vidas, T., Owusu, E., Wang, S., Zeng, C., Cranor, L.F., Christin, N.: Qrishing: The susceptibility of smartphone users to QR code phishing attacks. In: International Conference on Financial Cryptography and Data Security, pp. 52–69 (2013). Springer

[4] Sharevski, F., Devine, A., Pieroni, E., Jachim, P.: Phishing with malicious QR codes. In: Proceedings of the 2022 European Symposium on Usable Security, pp. 160–171 (2022)

[5] Al-Mannai, E.M., Litayem, N., Belhi, A., Al Samara, M.: Feature-based machine learning for real-time detection of malicious QR and NFC content. In: 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET), pp. 1–6 (2026). IEEE

[6] Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)

[7] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 3971–3988 (2022)

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

[9] Prasad, A., Chandra, S.: PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. Computers & Security **136**, 103545 (2024)

[10] Dalton, T., Gowda, H., Rao, G., Pargi, S., Khodabakhshi, A.H., Rombs, J., Jou, S., Marwah, M.: Phreshphish: A real-world, high-quality, large-scale phishing website dataset and benchmark. arXiv preprint arXiv:2507.10854 (2025)

[11] Lundberg, S.M., Lee, S.-I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)

[12] Krombholz, K., Frühwirt, P., Kieseberg, P., Kapsalis, I., Huber, M., Weippl, E.: QR code security: A survey of attacks and challenges for usable security. In: International Conference on Human Aspects of Information Security, Privacy, and Trust, pp. 79–90 (2014). Springer

[13] Onumadu, P., Abroshan, H.: Near-field communication (NFC) cyber threats and mitigation solutions in payment transactions: a review. Sensors **24**(23), 7423 (2024)

[14] Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M., Weippl, E.: QR code security. In: Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia, pp. 430–435 (2010)

[15] Geisler, M., Pöhn, D.: Hooked: a real-world study on QR code phishing. arXiv preprint arXiv:2407.16230 (2024)

[16] Han, X., Zhang, Y., Zhang, X., Chen, Z., Wang, M., Zhang, Y., Ma, S., Yu,

Y., Bertino, E., Li, J.: Medusa attack: Exploring security hazards of In-App QR code scanning. In: 32nd USENIX Security Symposium (USENIX Security 23), pp. 4607–4624 (2023)

[17] Alexiou, N., Basagiannis, S., Petridou, S.: Formal security analysis of near field communication using model checking. Computers & Security **60**, 1–14 (2016)

[18] Roland, M., Langer, J., Scharinger, J.: Applying relay attacks to Google Wallet. In: 2013 5th International Workshop on Near Field Communication (NFC), pp. 1–6 (2013). IEEE

[19] Miller, C.: Exploring the NFC attack surface. Proceedings of Blackhat (2012)

[20] Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254 (2009)

[21] Sahingoz, O.K., Buber, E., Demir, O., Diri, B.: Machine learning based phishing detection from URLs. Expert Systems with Applications **117**, 345–357 (2019)

[22] Do Xuan, C., Nguyen, H.D., Tisenko, V.N.: Malicious URL detection based on machine learning. International Journal of Advanced Computer Science and Applications **11**(1) (2020)

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

[24] Szurdi, J., Kocso, B., Cseh, G., Spring, J., Felegyhazi, M., Kanich, C.: The long "Taile" of typosquatting domain names. In: 23rd USENIX Security Symposium (USENIX Security 14), pp. 191–206. USENIX Association, Berkeley, CA, USA (2014)

[25] Trad, F., Chehab, A.: Detecting quishing attacks with machine learning techniques through QR code analysis. arXiv preprint arXiv:2505.03451 (2025)

[26] Sarkhi, M., Mishra, S.: Detection of QR code-based cyberattacks using a lightweight deep learning model. Engineering, Technology & Applied Science Research **14**(4), 15209–15216 (2024)

[27] Vishwakarma, P.P., Tripathy, A.K., Vemuru, S.: An empiric path towards fraud detection and protection for NFC-enabled mobile payment system. TELKOMNIKA (Telecommunication Computing Electronics and Control) **17**(5), 2313–2320 (2019)

[28] Tagle, R.A.: Machine learning integration for real-time fraud detection in near

field communication (NFC) card transactions. TECHNOLOGIQUE: A GLOBAL JOURNAL ON TECHNOLOGICAL DEVELOPMENTS AND SCIENTIFIC INNOVATIONS: VMC Analytiks Multidisciplinary Journal News Publishing Services **3**(1), 69–76 (2024)

[29] Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)

[30] Ghaleb Al-Mekhlafi, Z., Abdulkarem Mohammed, B., Al-Sarem, M., Saeed, F., Al-Hadhrami, T., Alshammari, M.T., Alreshidi, A., Sarheed Alshammari, T.: Phishing websites detection by using optimized stacking ensemble model. Computer Systems Science and Engineering **41**(1), 109–125 (2021)

[31] Jain, A., Khan, S., Koli, K., Taneja, H., Panwar, A., Alshammari, T., Obaid, E.M., Abdu, S.M.: Phishnet 1.0: optuna-optimized stacking ensemble with boruta-based feature selection for phishing URL detection. Scientific Reports **16**(1), 1480 (2026)

[32] Matole, D., Ajankar, S., Dutta, T., Mishra, R., Korde, L.: SHAP-driven feature selection with meta-ensemble classifiers for efficient phishing URL detection. In: International Conference on Intelligent Vision and Computing, pp. 337–349 (2025). Springer

[33] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: TESSERACT: Eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium (USENIX Security 19), pp. 729–746. USENIX Association, Berkeley, CA, USA (2019)

[34] Misiek, M., Hyla, T.: X-phide Phishing and Legitimate URLs with Features. https://doi.org/10.21227/7k88-0x11

[35] Mia, M., Derakhshan, D., Pritom, M.M.A.: Can features for phishing URL detection be trusted across diverse datasets? a case study with explainable ai. In: Proceedings of the 11th International Conference on Networking, Systems, and Security, pp. 137–145 (2024)

[36] Rashid, F., Doyle, B., Han, S.C., Seneviratne, S.: Phishing URL detection generalisation using unsupervised domain adaptation. Computer Networks **245**, 110398 (2024)

[37] Subairu, S., Alhassan, J., Abdulhamid, S., Ojeniyi, J.: A review of detection methodologies for quick response code phishing attacks. In: 2020 2nd International Conference on Computer and Information Sciences (ICCIS), pp. 1–5 (2020). IEEE

[38] samahsadiq: Benign and Malicious URLs. Kaggle dataset, https://www.kaggle.com/datasets/samahsadiq/benign-and-malicious-urls. Accessed September 2026 (2023)

[39] Kaitholikkal, J.K.S.: Phishing URL dataset. Mendeley Data, V1 (2024). https://doi.org/10.17632/vfszbj9b36.1

[40] sahandnamvar: Phishing URL Detection (111K URLs, 22 Features). Kaggle dataset, https://www.kaggle.com/datasets/sahandnamvar/phishing-url-detection-111k-urls-22-features. Accessed September 2026 (2025)