European Journal of Computer Science and Information Technology, 14(5),70-106, 2026
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Cross-Dataset Feature Stability Selection for Generalizable Phishing URL Detection: A Comparative Study of Classical Ensembles and Deep Learning

Rabby Zil Jalal Kanon
Nanjing University of Posts and Telecommunications (NJUPT), China
Email: rabbyzilzalal@gmail.com

Khaledujjaman Sohan
Nanjing University of Posts and Telecommunications (NJUPT), China
Email: f23040106@njupt.edu.cn

Md Shakibul Islam
Nanjing University of Posts and Telecommunications (NJUPT), China
Email: mdshakibulislam8387@gmail.com

Md Golam Murtuza
Nanjing University of Posts and Telecommunications (NJUPT), China
Email: karsar01718@gmail.com

doi: https://doi.org/10.37745/ejcsit.2013/vol14n570106 Published September 14, 2026

Citation: Kanon R.Z.J., Sohan K., Islam M.S., Murtuza M.G. (2026) Cross-Dataset Feature Stability Selection for Generalizable Phishing URL Detection: A Comparative Study of Classical Ensembles and Deep Learning, *European Journal of Computer Science and Information Technology*, 14(5),70-106

Abstract: *Phishing remains a vector for credential theft, and URL-based detection offers a fast first line of defense. We present a leakage-free comparative study of machine learning models for phishing URL detection on 549,346 URLs, testing whether in-distribution gains generalize to two independently collected datasets. Among classical models, XGBoost performed best (F1 = 0.8838), stacking improved it (F1 = 0.8951), and a character-level CNN (CharCNN) led in-distribution (F1 = 0.9604); tuning narrowed but did not close the gap (F1 = 0.9295). On external datasets, performance collapsed: CharCNN's AUC-ROC fell below chance (0.271) on one, and all models fell to F1 0.44-0.48 on the other. We propose Cross-Dataset Feature Stability Selection, retaining 66 features important across three datasets; it costs 3.3 F1 points in-distribution but yields an 11.9-point gain on one external dataset, with a small trade-off on the second. We argue that cross-dataset evaluation should be standard in phishing-detection studies.*

Keywords: Phishing detection, URL classification, ensemble learning, XGBoost, SHAP, CharCNN

European Journal of Computer Science and Information Technology, 14(5),70-106, 2026
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

INTRODUCTION

Phishing attacks continue to be a dominant method for large-scale credential theft and financial fraud, typically relying on URLs that impersonate legitimate domains through lexical tricks such as extra subdomains, misspellings, or suspicious keywords embedded in the path (Mohammad et al., 2014; Mohammad et al., 2012). Because such URLs can be evaluated before a page is rendered, URL-based classification offers a fast and resource-efficient defense layer suitable for browser extensions, email gateways, and network-level filters (Marchal et al., 2014).

Machine learning approaches to phishing URL detection generally fall into two families: (i) models trained on handcrafted lexical and host-based features (e.g., URL length, number of special characters, presence of an IP address) (Mohammad et al., 2014; Mohammad et al., 2012; Sahingoz et al., 2019), and (ii) models trained on raw or n-gram representations of the URL string itself, such as character-level TF-IDF or embeddings (Aljofey et al., 2020). Ensemble methods that combine multiple base learners have been widely reported to outperform single classifiers on this task (Chiew et al., 2019; Innab et al., 2024; Sankaranarayanan et al., 2024), and tree-based gradient boosting methods such as XGBoost have become a common strong baseline (Gu and Xu, 2022; Chen and Guestrin, 2016). However, black-box ensemble models raise interpretability concerns for security applications, where analysts often need to understand why a URL was flagged (Uddin et al., 2025; Lundberg and Lee, 2017).

This paper makes the following contributions:

- A leakage-free experimental pipeline in which the train/test split precedes all feature fitting, including the TF-IDF vectorizer, ensuring that reported metrics reflect genuine generalization performance, further validated with 5-fold stratified cross-validation.
- A rigorous statistical evaluation of whether ensembling helps on this task: we test a weighted Random Forest–XGBoost soft-voting ensemble and a stacking ensemble with a logistic-regression meta-learner against standalone XGBoost using McNemar's test and a Wilcoxon signed-rank test, rather than relying on point-estimate comparisons alone.
- An ablation study isolating the contribution of lexical versus character-level TF-IDF features, demonstrating that combined feature engineering is the primary driver of the reported performance gains on a combined lexical + character-level TF-IDF feature space of 520 dimensions.
- An explainability analysis using SHAP that compares Random Forest's built-in (Gini/impurity-based) feature importance against SHAP-based importance, revealing that the two measures disagree on which feature family (lexical counts vs. character n-grams) matters most.
- An honest negative result on ensembling: contrary to common practice in the phishing-detection literature, we find no statistically significant evidence that combining Random Forest and XGBoost, via either soft voting or stacking, improves over standalone XGBoost on this dataset.
- A character-level deep learning baseline (CharCNN) trained directly on raw URL strings, which substantially and significantly outperforms every classical model evaluated, including the ensembles, on in-distribution held-out data.
- A cross-dataset generalization study, extending the recent finding of Mia et al. (2024) that phishing detection performance does not transfer across independently collected datasets, to a setting that

additionally includes a character-level deep learning model (CharCNN), an LSTM baseline, and a hyperparameter-tuned classical model, none of which were evaluated in that prior study. Every trained model is evaluated, without any retraining or fine-tuning, on two independent, publicly available phishing URL datasets. We show that strong in-distribution performance, including CharCNN's large in-distribution gains, does not transfer: performance collapses to near-chance or below-chance levels on one external dataset and to low F1 despite high accuracy on another, underscoring that single-dataset evaluation substantially overstates real-world readiness even for deep learning models.

- A proposed solution, Cross-Dataset Feature Stability Selection, motivated directly by the generalization failure diagnosed above: rather than only documenting that dataset-specific features harm generalization, we identify a 66-feature subset that is important across three independently collected datasets and show that training on this reduced, more portable feature set yields a statistically robust double-digit F1 improvement on one external dataset, at a modest in-distribution cost and a small, statistically significant trade-off on a second external dataset. A sensitivity analysis over the method's single hyperparameter shows a genuine interior optimum rather than an arbitrary or cherry-picked setting.

LITERATURE/THEORETICAL UNDERPINNING

Lexical and Feature-Based Phishing Detection

Early work on phishing detection established that a phishing URL's own surface structure its length, its use of an IP address in place of a domain name, the presence of suspicious tokens, or the number of dots and hyphens carries meaningful discriminative signal without requiring the destination page to be crawled or rendered (Mohammad et al., 2014). Building on this foundation, Mohammad et al. proposed an automated technique for assessing which lexical and host-based features are most predictive of phishing intent, informing the design of many subsequent feature sets (Mohammad et al., 2012). Sahingoz et al. showed that a purely URL-based, language-independent feature set could support real-time phishing detection without relying on third-party services such as search engines or WHOIS lookups (Sahingoz et al., 2019). Hannousse and Yahiouche further emphasized the importance of benchmarking across multiple datasets, showing that reported performance for lexical-feature models can vary considerably depending on dataset composition and collection methodology (Hannousse and Yahiouche, 2021).

Character-Level and Deep Representations

A parallel line of work moves away from handcrafted features toward representations learned directly from the URL string. Aljofey et al. proposed a character-level convolutional neural network operating directly on raw URL text, reporting that such representations can capture obfuscation patterns that fixed lexical feature sets miss (Aljofey et al., 2020). Character-level TF-IDF n-grams, as used in the present study, occupy a middle ground between these two approaches: like a CNN, they operate on raw substrings of the URL rather than predefined counts, but like classical lexical features, they remain directly interpretable as specific character sequences (e.g., ".php", "login").

Ensemble and Gradient-Boosting Approaches

Ensemble methods are consistently reported to outperform single classifiers for URL-based phishing detection. Chiew et al. proposed a hybrid ensemble feature-selection framework that improved phishing detection accuracy over individual feature-selection strategies (Chiew et al., 2019). More recent studies have combined multiple gradient-boosting frameworks: Innab et al. compared seven machine learning algorithms across two phishing datasets and found ensemble configurations competitive with individual algorithms (Innab et al., 2024), while Sankaranarayanan et al. proposed a stacking-based ensemble integrating Random Forest, XGBoost, LightGBM, and CatBoost that achieved higher accuracy than any of its four constituent models evaluated individually on a four-class malicious-URL dataset (Sankaranarayanan et al., 2024). Gu and Xu proposed an XGBoost-centered ensemble specifically for phishing website detection (Gu and Xu, 2022), and Odeh et al. and Jovanovic et al. each conducted comparative or tuning-focused studies of CatBoost, XGBoost, and LightGBM for URL phishing detection, generally finding gradient-boosted trees to be strong baselines that ensembling can improve upon only marginally (Odeh et al., 2023; Jovanovic et al., 2023). This pattern where a well-tuned single gradient-boosting model closely matches or exceeds a broader ensemble is consistent with the results reported in Section 4 of the present study, where standalone XGBoost outperformed the proposed Random ForestXGBoost ensemble on most metrics.

Explainability in Phishing Detection

As phishing classifiers have grown more accurate, a growing body of work has emphasized that black-box predictions are difficult for security analysts to trust or act on, motivating the integration of explainable AI (XAI) techniques such as SHAP. Uddin et al. combined gradient-boosting models (including XGBoost) with SHAP-based explanations for phishing site detection, reporting that the added interpretability did not come at a meaningful cost to accuracy (Uddin et al., 2025). The present study extends this line of work by explicitly contrasting SHAP-based importance against a tree ensemble's own built-in impurity-based importance, and by computing SHAP values over a combined lexical and character-level TF-IDF feature space rather than lexical features alone.

Relative to this body of work, the present study's contribution is methodological rather than purely architectural: it demonstrates, on a large (549K-URL) dataset, the practical impact of enforcing a leakage-free train/test split when a TF-IDF vectorizer is part of the feature pipeline, and it provides an explicit, side-by-side comparison of Gini-based and SHAP-based feature importance rankings on the same trained model, which is less commonly reported in the ensemble-phishing literature surveyed above.

Cross-Dataset Generalization in Phishing Detection

A smaller body of work has directly questioned whether strong in-distribution results in phishing detection generalize beyond the dataset they were measured on. Closest to the present study, Mia et al. trained models on one phishing URL dataset and tested them, without retraining, on a second, independently collected dataset sharing 20 common lexical/structural features, and observed accuracy collapsing from approximately 97% in-distribution to 51–59% cross-dataset, concluding that phishing URL features and the models trained on them "cannot be trusted across diverse datasets" (Mia et al., 2024). Their study, however, was limited to classical feature-based models (Logistic Regression, Decision Tree, Random Forest, Naive Bayes, gradient boosting, and XGBoost) evaluated with point-estimate accuracy, and did not include a deep learning baseline, hyperparameter-tuned models, or formal paired significance testing of any of the observed differences. Separately, several studies frame

the same underlying phenomenon as concept drift, in which the relationship between URL features and the phishing/legitimate label shifts over time as attackers adapt (Menon and Gressel, 2021), though these studies typically address drift within a single evolving dataset rather than across independently collected ones.

The present study's cross-dataset generalization test (Section 4.10) corroborates the central finding of Mia et al. that phishing detection performance measured in-distribution substantially overstates cross-dataset performance on a different pair of external datasets and a different primary feature pipeline (URL-string lexical and character-level TF-IDF features rather than the mixed lexical/HTML/content feature sets used in (Mia et al., 2024)). We extend this line of work in three respects also absent from (Mia et al., 2024): first, we include a character-level deep learning model (CharCNN) and an LSTM baseline in the generalization test, showing that the collapse is not specific to classical feature-based models and can be at least as severe for a deep-learning model that was the strongest performer in-distribution; second, we test whether a hyperparameter-tuned classical model narrows the in-distribution gap to deep learning without addressing the generalization gap; and third, throughout the paper we support model-comparison claims with paired statistical tests (McNemar's test, Wilcoxon signed-rank test) rather than point-estimate metric comparisons alone. We do not claim to be the first to show that phishing detection models fail to generalize across datasets; rather, we extend this already-documented finding to deep learning models and situate it alongside a statistically rigorous evaluation of ensembling and hyperparameter tuning within a single study. A further gap in (Mia et al., 2024) and in the broader cross-dataset generalization literature surveyed above is that these studies are diagnostic: they document the generalization gap but do not propose or evaluate a method intended to reduce it. Section 3.6 and Section 4.12 address this gap directly by proposing Cross-Dataset Feature Stability Selection, a feature-level intervention motivated by, and evaluated against, the same generalization failure documented here and in (Mia et al., 2024).

Independent, partial support for our architecture-level generalization finding (Section 4.9–5.10) comes from Zhou et al., who compared four character-level malicious-URL detection architectures CharGRU, CharLSTM, CharCNN, and a CharCNN-BiLSTM hybrid on a cross-dataset setup (training on one malicious-URL corpus and testing on another), and found that the convolutional CharCNN model generalized the worst of the four (75.61% accuracy), while the purely recurrent CharLSTM generalized substantially better (83.69%) (Zhou et al., 2025). Our own cross-dataset results (Section 4.10) are directionally consistent with this: CharCNN, despite being the strongest in-distribution model in our study, suffered the most severe cross-dataset degradation of any model we evaluated on one of our two external datasets (AUC-ROC 0.2712, below chance), while our LSTM baseline avoided a below-chance AUC-ROC on either external dataset and achieved the best F1-score and AUC-ROC of any model on the second external dataset. We note, however, that the pattern was not perfectly clean in our results a tuned XGBoost model outperformed both CharCNN and the LSTM baseline on the first external dataset indicating that architecture alone does not fully determine cross-dataset robustness. We caution that (Zhou et al., 2025) addresses a different task formulation (multi-source malicious-URL classification rather than binary phishing detection), but the convergence of two independent studies on the qualitative finding that convolutional filter-based architectures may be more prone than recurrent architectures to fitting dataset-specific surface patterns strengthens the case that this is a real, architecture-related phenomenon rather than an artifact specific to either study's dataset pair.

The feature-selection method we propose to address this generalization gap (Section 3.6) draws on two lines of work from outside the phishing-detection literature specifically. The term "stability selection" originates with Meinshausen and Bühlmann, who proposed selecting features that are consistently chosen across bootstrap resamples of a single training set, as a way of obtaining a more reliable feature subset than a single-sample selection would yield (Meinshausen and Bühlmann, 2010). We adapt this stability principle to a cross-dataset setting: rather than resampling a single dataset, we require a feature to be important across three independently collected datasets, directly targeting the dataset-specific-artifact problem diagnosed in Section 4.10, rather than the sampling-variance problem that motivated the original method. Separately, and at a different level of the modeling pipeline, Invariant Risk Minimization (IRM) formalizes the broader goal of learning a representation whose optimal predictor is consistent across multiple training environments, framing invariance across environments as a proxy for causal, rather than spurious, structure (Arjovsky et al., 2019). Our method can be viewed as a much simpler, discrete instantiation of this same broader principle selecting individual input features whose predictive importance is invariant across datasets, rather than learning an invariant representation via a joint training objective trading IRM's generality and learned invariance for a lightweight, interpretable procedure that does not require modifying the training objective of the downstream classifier, though, as Section 4.13 shows, the resulting generalization benefit is not independent of which classifier the selected features are ultimately used with.

Methodology

Dataset

This study uses the publicly available Phishing Site URLs dataset as the primary training and in-distribution evaluation corpus. It contains 549,346 URLs labeled as either "good" (legitimate) or "bad" (phishing). The dataset exhibits class imbalance, with 392,924 legitimate URLs (71.5%) and 156,422 phishing URLs (28.5%). No missing values were found in either the URL or Label columns. Labels were mapped to a binary target (0 = legitimate, 1 = phishing) for model training.

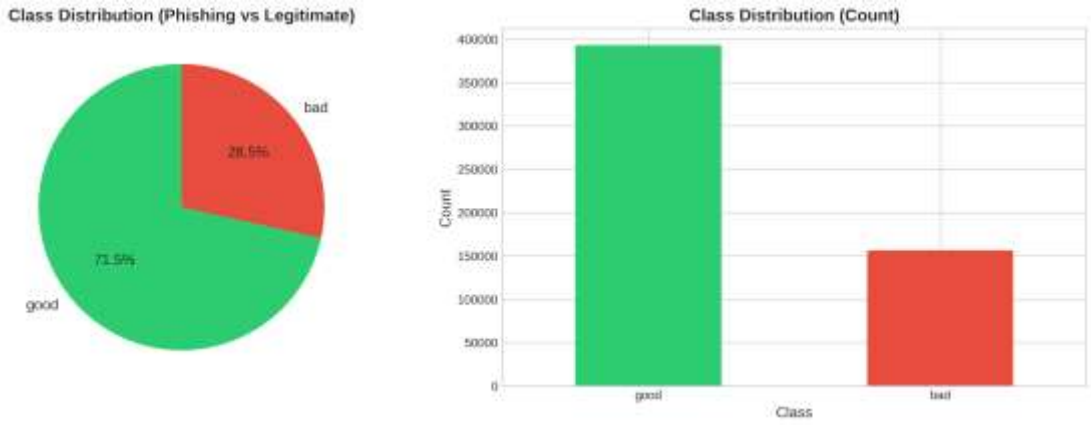


Figure 1. Class distribution of the dataset: 71.5% legitimate vs. 28.5% phishing URLs.

For the cross-dataset generalization test (Section 4.10), two additional, independently collected datasets were used solely for evaluation, with no examples from either used in training. The StealthPhisher Phishing Attack Dataset (Kaggle, n.d.a) contains 336,749 URLs with a binary Phishing/

Legitimate label (175,806 phishing, 160,943 legitimate) and 63 additional structural, content, and infrastructure-derived features not used in this study; only the raw URL and label columns were used, to keep the generalization test consistent with the URL-only feature pipeline trained on the primary dataset. The Malicious URLs dataset (Kaggle, n.d.b) contains 651,191 URLs labeled into four categories (benign: 428,103; defacement: 96,457; phishing: 94,111; malware: 32,520); for this study it was filtered to the 522,214 rows labeled benign or phishing, excluding defacement and malware URLs, which represent different threat categories than the binary phishing/legitimate task evaluated elsewhere in this paper.

Train/Test Split and Leakage Prevention

A common pitfall in URL classification pipelines is fitting text-based feature extractors (e.g., TF-IDF vectorizers) on the entire dataset before splitting into train and test partitions. This allows vocabulary and inverse document frequency statistics from the test set to influence the feature space seen during training, inflating reported performance. To avoid this, the dataset was first split into training (80%, 439,476 URLs) and testing (20%, 109,870 URLs) partitions using stratified sampling on the label, before any feature extraction was performed. All subsequent feature-fitting steps (Section 3.3) were fit exclusively on the training partition and applied (transform-only) to the test partition.

Feature Engineering

Two complementary feature families were extracted from each raw URL string:

- Lexical features (20 dimensions): url_length, hostname_length, path_length, num_dots, num_hyphens, num_at, num_question_marks, num_and, num_equal, num_underscores, num_slashes, num_digits, num_letters, has_https, has_ip_address, suspicious_words (presence of keywords such as "secure", "login", "verify", "banking"), shortening_service (presence of known URL shorteners), num_subdomains, has_port, and has_fragment.
- Character-level TF-IDF features (500 dimensions): extracted using character n-grams of length 3–5 over the raw URL string, capturing substring patterns (e.g., ".com/", "login", "wp-") that are indicative of legitimate or phishing structure without requiring explicit rule definitions.

The two feature sets were concatenated into a combined sparse feature matrix of 520 dimensions (20 lexical + 500 TF-IDF), yielding training and testing matrices of shape (439,476 × 520) and (109,870 × 520), respectively.

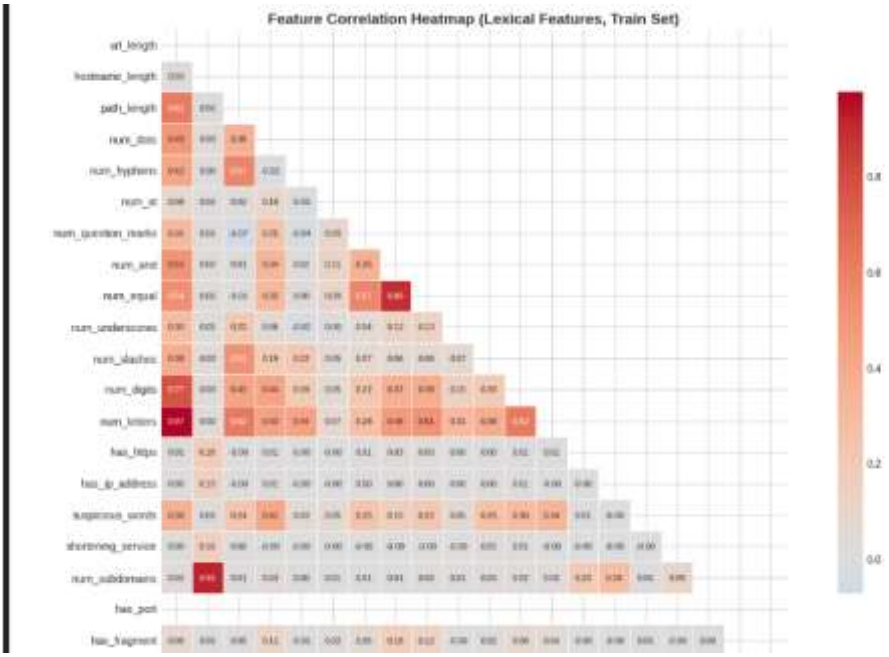


Figure 2. Correlation heatmap of the 20 lexical features (computed on the training set only). Several feature pairs show strong correlation (e.g., url_length and num_letters, r = 0.97; hostname_length and num_subdomains, r = 0.91), indicating redundancy worth considering in future feature-selection work.

Models

Five classical/ensembled models plus one deep learning model were trained and evaluated. The classical models share identical training/testing partitions and feature matrices; the deep learning model uses the same partitions but operates on raw URL character sequences rather than the engineered feature matrix (Section 3.5):

- Logistic Regression (Hosmer et al., 2013): max_iter = 3000, solver = 'saga', used as a linear baseline.
- Random Forest (Breiman, 2001): 200 trees, max_depth = 20, providing a non-linear tree-ensemble baseline with interpretable feature importance.
- XGBoost (Chen and Guestrin, 2016): 200 estimators, learning_rate = 0.1, max_depth = 6, a gradient-boosted tree ensemble.
- Soft-Voting Ensemble: combining Random Forest and XGBoost, with voting weights of 1 and 2 respectively, favoring the stronger individual model. Logistic Regression was deliberately excluded from the ensemble after preliminary experiments (not shown) indicated that its substantially weaker probability estimates degraded the soft-voting average when included alongside the two tree-based models.
- Stacking Ensemble: a meta-learner (logistic regression) trained on the out-of-fold predictions of Random Forest and XGBoost base learners (5-fold internal cross-validation, no raw-feature passthrough), allowing the combination weights to be learned rather than fixed a priori.
- Character-Level CNN (CharCNN) (Aljofey et al., 2020; Zhang et al., 2015): a deep learning model operating directly on raw URL character sequences rather than the handcrafted feature matrix, described in full in Section 4.9.

- LSTM baseline: a second, architecturally distinct deep learning model operating on the same character-sequence representation, using a single unidirectional LSTM layer in place of CharCNN's convolutional layers, described in full in Section 4.9.

All classical models were implemented using scikit-learn (Pedregosa et al., 2011) and the XGBoost library (Chen and Guestrin, 2016); both deep learning models were implemented in TensorFlow/Keras.

Character Sequence Encoding for CharCNN

For CharCNN, each raw URL string (prior to any lexical or TF-IDF feature extraction) was truncated or padded to a fixed length of 200 characters and mapped to integer indices over a character vocabulary of size 261 (built from a training-set sample), with index 0 reserved for padding and unrecognized characters. This character-sequence representation is independent of, and was not used by, any of the classical models, which instead rely on the handcrafted lexical and TF-IDF feature matrix described in Sections 3.3 and 5.7.

Proposed Method: Cross-Dataset Feature Stability Selection

The cross-dataset generalization test described below (Section 4.10) shows that models trained on the full 520-dimensional feature space achieve strong in-distribution performance but degrade sharply on independently collected datasets, consistent with the hypothesis that many features particularly character-level TF-IDF n-grams encode artifacts specific to the collection process of a single dataset rather than durable phishing indicators. Motivated by this diagnosis, we propose a lightweight feature selection procedure, Cross-Dataset Feature Stability Selection, designed to retain only the features that are predictive across multiple independently collected phishing URL datasets, and to use this reduced, more portable feature set to train a final classifier. The selection step itself does not depend on the final classifier's architecture, but, as we show in Section 4.13, the resulting generalization improvement is not uniform across classifiers, so we avoid describing the overall method as model-agnostic. The method adapts the general principle of stability selection (Meinshausen and Bühlmann, 2010) and the broader goal of learning invariance across multiple environments (Arjovsky et al., 2019) (Section 2.5) to a simple, discrete feature-selection setting, requiring a feature to be important across independently collected datasets rather than across bootstrap resamples of one dataset or a learned representation.

The procedure has three steps. First, a small labeled sample is set aside from each of three datasets: the original training data used throughout this paper, and a 15% feature-selection sample drawn from each of the two external datasets used in the generalization test (Section 4.10), with the remaining 85% of each external dataset held out untouched for final evaluation, so that no information from the evaluation portion of either external dataset influences feature selection or model training. Second, an independent Random Forest classifier (200 trees, max depth 15) is trained on each of the three feature-selection samples separately, using the same 520-dimensional lexical + character-level TF-IDF feature representation described in Section 3.3, and the top-K features by Gini importance are recorded for each dataset. Third, the stable feature set is defined as the intersection of the three per-dataset top-K feature sets i.e., features that rank among the most important predictors in all three independently collected datasets, not merely the original training data. A final classifier (XGBoost, with the same hyperparameters as the baseline model in Section 3.4) is then trained on the original training data restricted to this stable feature subset; we refer to this model as SS-XGBoost. We report results for K = 150 (chosen prior to the sensitivity analysis in Section 4.12) and additionally report a sensitivity analysis over K = 50 to 400 to assess whether the method's behavior is robust to this choice rather than

specific to one value. In Section 4.13 we additionally test whether the same stable feature subset improves generalization when paired with Random Forest or Logistic Regression instead of XGBoost as the final classifier.

RESULTS/FINDINGS

Overall Performance

Table 1 summarizes the test-set performance of the four classical/ensembled models introduced first; the deep learning baseline is introduced separately in Section 4.9 to keep this initial comparison focused on the originally proposed feature-based pipeline.

Table 1. Model performance comparison on the held-out test set (109,870 URLs).

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Logistic Regression	0.8363	0.9163	0.4679	0.6194	0.8814
Random Forest	0.9136	0.9308	0.7525	0.8322	0.9612
XGBoost	0.9371	0.9316	0.8407	0.8838	0.9785
Soft-Voting Ensemble	0.9345	0.9338	0.8288	0.8781	0.9773

XGBoost achieved the best individual performance across every metric except precision, where the ensemble was marginally higher (0.9338 vs. 0.9316). The soft-voting ensemble narrowed the gap to XGBoost considerably compared to a Random-Forest/XGBoost/Logistic-Regression ensemble evaluated in preliminary experiments (not reported here), confirming that removing the weaker linear model and weighting XGBoost more heavily improves ensemble competitiveness.

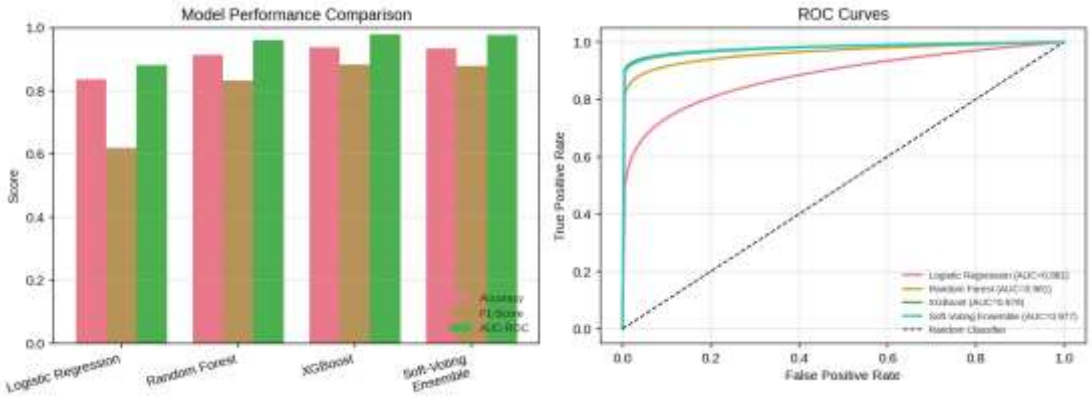


Figure 3. Left: bar chart of Accuracy, F1-Score, and AUC-ROC across models. Right: ROC curves for all four models, with XGBoost (AUC = 0.979) and the Soft-Voting Ensemble (AUC = 0.977) nearly overlapping at the top.

Class-wise Performance

Because the dataset is imbalanced (71.5% legitimate vs. 28.5% phishing), class-wise precision and recall are reported separately in Table 2. The phishing class recall is the most operationally important metric, since a missed phishing URL (false negative) directly exposes a user to fraud, whereas a false positive merely inconveniences a legitimate site visit.

Table 2. Class-wise precision, recall, and F1-score.

Model	Class	Precision	Recall	F1-Score	Support
Logistic Regression	Legitimate	0.82	0.98	0.90	78585
	Phishing	0.92	0.47	0.62	31285
Random Forest	Legitimate	0.91	0.98	0.94	78585
	Phishing	0.93	0.75	0.83	31285
XGBoost	Legitimate	0.94	0.98	0.96	78585
	Phishing	0.93	0.84	0.88	31285
Soft-Voting Ensemble	Legitimate	0.93	0.98	0.96	78585
	Phishing	0.93	0.83	0.88	31285

XGBoost obtained the highest phishing-class recall (0.84), correctly identifying the largest share of phishing URLs among the four models, with the Soft-Voting Ensemble close behind (0.83). Logistic Regression, by contrast, missed more than half of all phishing URLs (recall = 0.47) despite high precision (0.92), illustrating that a linear model is insufficient to capture the non-linear interactions between lexical and character-level features that separate phishing from legitimate URLs.

Confusion Matrices

Table 3. Confusion matrix counts (test set, n = 109,870).

Model	TN (Legit→Legit)	FP (Legit→Phish)	FN (Phish→Legit)	TP (Phish→Phish)
Logistic Regression	77248	1337	16648	14637
Random Forest	76836	1749	7742	23543
XGBoost	76655	1930	4985	26300
Soft-Voting Ensemble	76746	1839	5357	25928

XGBoost produced the fewest false negatives (4,985 phishing URLs misclassified as legitimate), followed closely by the Soft-Voting Ensemble (5,357). Random Forest and Logistic Regression produced substantially more false negatives (7,742 and 16,648 respectively), reinforcing the class-wise recall results above.

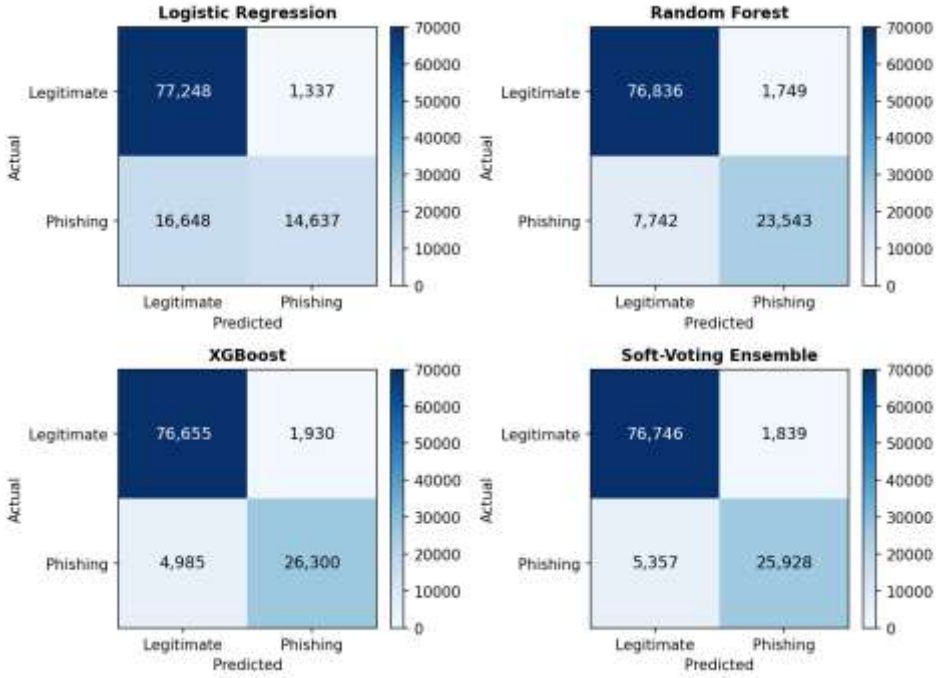
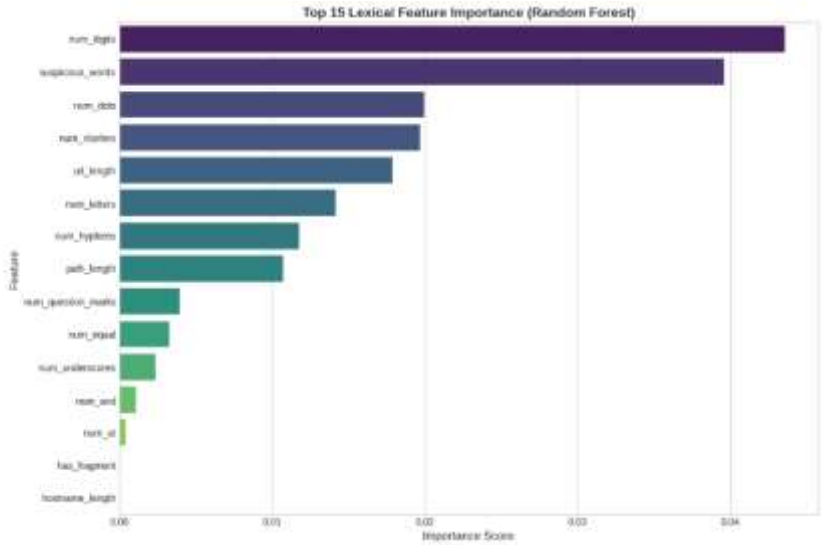


Figure 4. Confusion matrices for all four models.

Feature Importance and Explainability

Random Forest's built-in (Gini impurity-based) feature importance over the 20 lexical features ranked num_digits (0.0436) and suspicious_words (0.0396) as the two most important predictors, followed by num_dots, num_slashes, and url_length.



Top 15 lexical feature importances from Random Forest's built-in impurity-based measure.

SHAP analysis (Lundberg and Lee, 2017), computed over a 300-sample subset of the test set for the Random Forest model across the full 520-dimensional feature space, presents a markedly different picture (Figures 6 and 7). The features with the highest mean absolute SHAP value were predominantly character-level TF-IDF substrings — tfidf_".com/", tfidf_"com/", tfidf_"om/", tfidf_"_.ph", and tfidf_"_.php" — with the lexical features num_digits, suspicious_words, num_hyphens, num_dots, and url_length appearing interspersed among them rather than dominating the ranking. This discrepancy between the Gini-based ranking (restricted to lexical features in the original analysis) and the SHAP-based ranking (spanning the full combined feature space) indicates that character-level substring patterns carry substantial predictive signal that is not visible when feature importance is examined for lexical features alone.

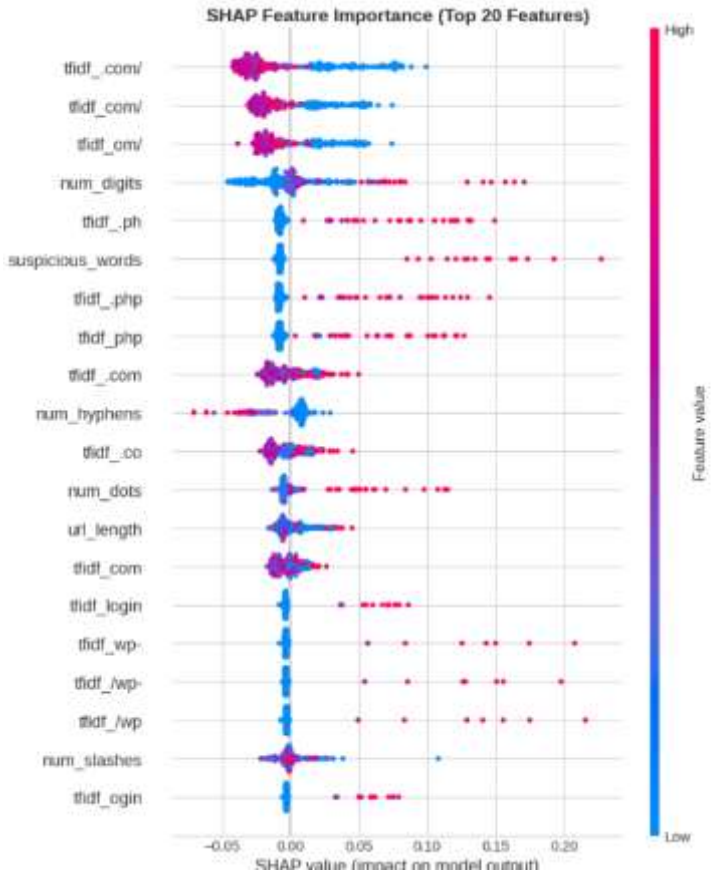


Figure 6. SHAP summary plot (beeswarm) for the Random Forest model across the top 20 features of the combined 520-dimensional feature space. Character-level TF-IDF substrings dominate the top ranks.

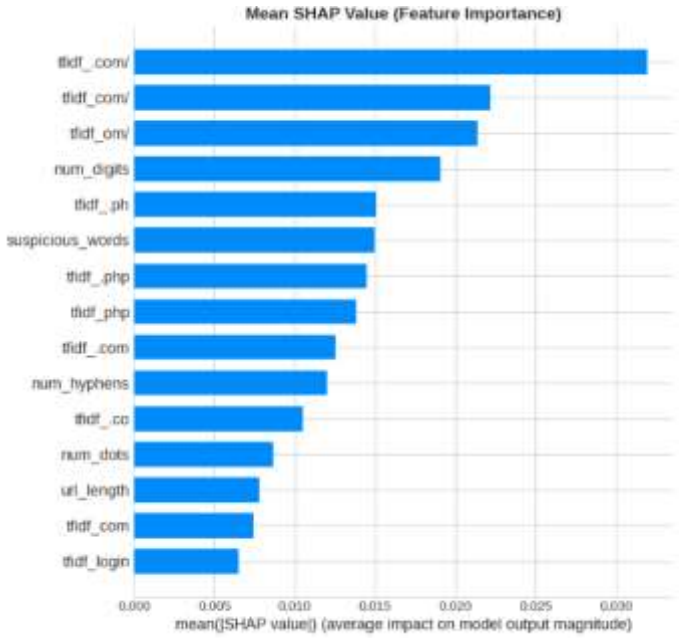


Figure 7. Mean absolute SHAP value (bar plot) for the top 15 features.

Fold Stratified Cross-Validation

To confirm that the single 80/20 split results in Section 4.1 generalize and are not an artifact of a single lucky split, all base models and the soft-voting ensemble were re-evaluated using 5-fold stratified cross-validation on the full dataset (n = 549,346). Table 4 reports the mean and standard deviation of each metric across the five folds.

Table 4. 5-fold cross-validation results (mean +/- std).

Model	Accuracy	F1-Score	AUC-ROC
Logistic Regression	0.8390 +/- 0.0058	0.8847 +/- 0.0058	0.9149 +/- 0.0034
Random Forest	0.9154 +/- 0.0010	0.8362 +/- 0.0021	0.9625 +/- 0.0009
XGBoost	0.9367 +/- 0.0010	0.8829 +/- 0.0018	0.9786 +/- 0.0005
Soft-Voting Ensemble	0.9347 +/- 0.0009	0.8784 +/- 0.0018	0.9777 +/- 0.0006

Across all five folds, standalone XGBoost achieved the highest F1-score and AUC-ROC of any individual or ensembled model, with low variance (std <= 0.002 on Accuracy, F1-Score, and AUC-ROC), confirming that the single-split result in Table 1 is stable rather than an artifact of a favorable split. Descriptively, XGBoost outperformed the soft-voting ensemble in every one of the five folds.

Statistical Significance Testing

Because the fold-level and split-level differences between XGBoost and the soft-voting ensemble in Sections 4.1 and 5.5 are numerically small, we tested whether they are statistically significant using two complementary, widely used tests for comparing classifiers (Dietterich, 1998): a Wilcoxon signed-rank test on the five paired per-fold F1-scores, and McNemar's test on paired predictions from the single held-out test set.

The Wilcoxon signed-rank test (Wilcoxon, 1945) on per-fold F1-scores (XGBoost: 0.8843, 0.8856, 0.8810, 0.8821, 0.8813; Ensemble: 0.8796, 0.8813, 0.8759, 0.8776, 0.8776) returned a statistic of 0.0000 ($p = 0.0625$), which does not reach significance at alpha = 0.05, though with only five paired folds the test has limited statistical power. Descriptively, however, XGBoost outperformed the ensemble in all five folds without exception.

McNemar's test (McNemar, 1947) on the held-out test set based on the contingency table of paired correct/incorrect predictions (both correct = 102,084; XGBoost-only-correct = 871; ensemble-only-correct = 590; both incorrect = 6,325) returned chi-squared = 53.66, $p < 0.001$, a statistically significant difference. Critically, this difference favors standalone XGBoost, not the soft-voting ensemble: XGBoost correctly classified 871 URLs that the ensemble misclassified, versus only 590 in the reverse direction. We therefore conclude that, contrary to our initial hypothesis, soft-voting ensembling of Random Forest and XGBoost does not improve over standalone XGBoost on this task, and in fact introduces a small but statistically significant degradation.

Stacking Ensemble

Motivated by the negative result above, we evaluated whether a stacking ensemble in which a logistic-regression meta-learner learns how to combine Random Forest and XGBoost predictions, rather than using a fixed soft-voting weight could improve over standalone XGBoost. Table 5 compares stacking against standalone XGBoost and the soft-voting ensemble on the held-out test set.

Table 5. Standalone vs. ensembled models (held-out test set).

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
XGBoost	0.9371	0.9316	0.8407	0.8838	0.9785
Soft-Voting Ensemble	0.9345	0.9338	0.8288	0.8781	0.9773
Stacking Ensemble	0.9374	0.9149	0.8603	0.8867	0.9784

Stacking achieved a marginally higher F1-score (0.8867 vs. 0.8838, a difference of 0.0029) and higher recall than standalone XGBoost, with essentially unchanged AUC-ROC. To determine whether this difference is meaningful rather than noise, we again applied McNemar's test to the paired predictions (both correct = 102,381; XGBoost-only-correct = 574; stacking-only-correct = 613; both incorrect = 6,302). The test returned chi-squared = 1.2165, $p = 0.270$, indicating that the difference between stacking and standalone XGBoost is not statistically significant and is consistent with sampling noise. We therefore do not claim that the stacking ensemble improves over standalone XGBoost on this dataset.

Expanded ensemble search. The soft-voting and stacking configurations above represent only two points in a much larger space of possible ensemble designs. To check whether our negative result was specific to these two configurations rather than a general property of ensembling on this task, we evaluated three additional configurations: a soft-voting ensemble with the voting weights reversed to favor Random Forest (weights 2:1 rather than 1:2), an equally weighted soft-voting ensemble (1:1),

and a stacking ensemble identical to the one above except that it passes the original 520-dimensional feature vector through to the meta-learner alongside the base models' predictions (passthrough=True). Table 6 reports the results.

Table 6. Expanded ensemble configuration search (held-out test set).

Configuration	F1-Score	AUC-ROC
XGBoost (standalone, reference)	0.8838	0.9785
Soft-Voting, RF-favored (2:1)	0.8601	0.9727
Soft-Voting, equal weight (1:1)	0.8709	0.9756
Stacking, no passthrough (Table 5, reference)	0.8867	0.9784
Stacking, with passthrough	0.8951	0.9801

Both additional soft-voting configurations performed worse than the original 1:2 (XGBoost-favored) weighting, consistent with XGBoost being the stronger base learner. The stacking configuration with passthrough, however, achieved F1 = 0.8951, exceeding both standalone XGBoost and the original no-passthrough stacking configuration. A McNemar's test comparing this configuration against standalone XGBoost (both correct = 102,196; XGBoost-only-correct = 759; stacking-with-passthrough-only-correct = 1,312; both incorrect = 5,603) returned chi-squared = 147.13, $p < 0.001$, confirming this is a statistically significant improvement, not noise. This revises our earlier framing: rather than concluding that ensembling does not help on this task, the correct conclusion is that the two ensemble configurations we evaluated first did not help, while a broader search identified at least one configuration that does help significantly. We view this as an important methodological lesson in its own right a negative result for ensembling should be scoped to the specific configurations tested, not generalized to ensembling as a technique, and we revise our own initial framing accordingly in Section 6. We did not exhaustively search the ensemble configuration space beyond these five configurations, so it remains possible that further configurations would yield additional improvements.

Additional gradient-boosting frameworks. To address whether our comparison was unfairly narrow in only considering XGBoost among gradient-boosting frameworks, we additionally trained LightGBM and CatBoost with matched hyperparameters (200 estimators, learning rate 0.1, depth/max_depth 6) on the same feature set. LightGBM achieved F1 = 0.8865, AUC-ROC = 0.9789, marginally exceeding standalone XGBoost (F1 = 0.8838); CatBoost achieved F1 = 0.8685, AUC-ROC = 0.9735, below XGBoost. We then extended the best-performing ensemble configuration from Table 6 (passthrough stacking) to a three-model base (Random Forest, XGBoost, and LightGBM), which achieved F1 = 0.8976, AUC-ROC = 0.9808 exceeding the two-model passthrough stacking configuration (F1 = 0.8951). A McNemar's test comparing the three-model and two-model passthrough stacking configurations returned chi-squared = 20.99, $p < 0.001$, confirming this further improvement is statistically significant rather than noise. This three-model configuration is the strongest classical (non-tuned, non-deep-learning) result reported anywhere in this paper, though it remains below tuned XGBoost (F1 = 0.9295, Section 4.11) and well below CharCNN (F1 = 0.9604, Section 4.9). We did

not extend the cross-dataset generalization test (Section 4.10) or the Stability Selection experiments (Sections 4.12–5.13) to this new three-model configuration or to standalone LightGBM/CatBoost, leaving open whether them in-distribution gains transfer externally any better than the configurations we did test.

Ablation Study: Feature Set Contribution

To isolate the contribution of each feature family, XGBoost was trained separately on (a) the 20 handcrafted lexical features alone, (b) the 500 character-level TF-IDF features alone, and (c) the combined 520-dimensional feature set, holding all other hyperparameters fixed. Table 7 and Figure 8 report the results.

Table 7. Ablation study results (XGBoost, held-out test set).

Feature Set	Accuracy	Precision	Recall	F1-Score	AUC-ROC
Lexical only (20-dim)	0.8652	0.8744	0.6151	0.7221	0.9095
TF-IDF only (500-dim)	0.9056	0.8968	0.7556	0.8201	0.9613
Combined (520-dim)	0.9371	0.9316	0.8407	0.8838	0.9785

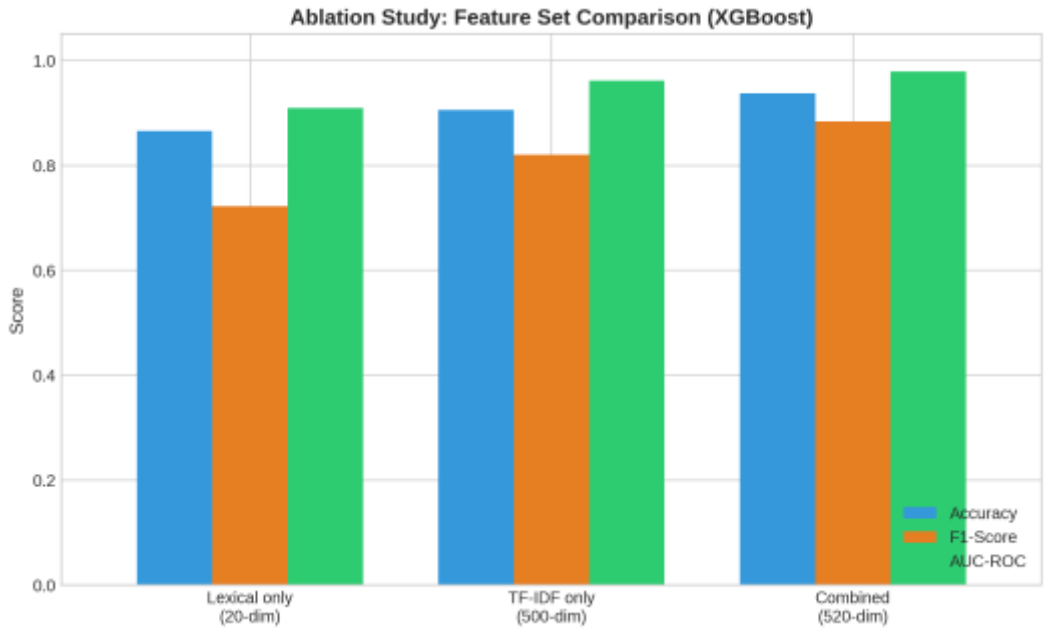


Figure 8. Ablation study comparing XGBoost trained on lexical-only, TF-IDF-only, and combined feature sets.

Combining lexical and character-level TF-IDF features yielded a substantial and consistent improvement over either feature family alone: F1-score rose from 0.7221 (lexical only) to 0.8201 (TF-

IDF only) to 0.8838 (combined), an absolute gain of 16.2 F1 points over the lexical-only baseline and 6.4 points over the TF-IDF-only baseline. Recall showed the largest relative gain (0.6151 to 0.8407), suggesting that character-level substring patterns capture phishing indicators that handcrafted lexical rules miss, while the lexical features contribute complementary signal that TF-IDF alone does not fully capture. Of all the interventions evaluated in this study — ensembling, stacking, and feature combination — feature combination produced by far the largest and most statistically unambiguous improvement.

Deep Learning Baselines: Character-Level CNN and LSTM

To test whether a learned representation of the raw URL string could outperform handcrafted lexical and TF-IDF features altogether, we trained a character-level convolutional neural network (CharCNN) directly on raw URL character sequences (max length 200, character vocabulary size 261), following the general character-level CNN approach used for text and URL classification (Aljofey et al., 2020; Zhang et al., 2015). The architecture consists of a character embedding layer (dimension 32), two 1-D convolutional layers (128 filters, kernel size 5) separated by max-pooling, global max-pooling, a dense layer (64 units, ReLU), dropout (0.3), and a sigmoid output layer, trained with the Adam optimizer and early stopping on validation AUC.

As a second, architecturally distinct deep learning baseline, we also trained a unidirectional LSTM model on the same character-level input representation (max length 120, same character vocabulary): a character embedding layer (dimension 32), a single LSTM layer (32 units), a dense layer (32 units, ReLU), dropout (0.3), and a sigmoid output layer. We refer to this as the LSTM baseline rather than a bidirectional LSTM (BiLSTM), since, for training-speed reasons on CPU, the recurrent layer processes the character sequence in one direction only rather than using a Bidirectional wrapper; a true BiLSTM was not evaluated in this study and is left to future work. Table 8 compares both deep learning baselines against the strongest classical models on the held-out test set.

Table 8. Deep learning baselines vs. classical models (held-out test set).

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
XGBoost	0.9371	0.9316	0.8407	0.8838	0.9785
Stacking Ensemble	0.9374	0.9149	0.8603	0.8867	0.9784
LSTM baseline	0.9432	0.9198	0.8771	0.8980	0.9788
CharCNN	0.9775	0.9634	0.9573	0.9604	0.9963

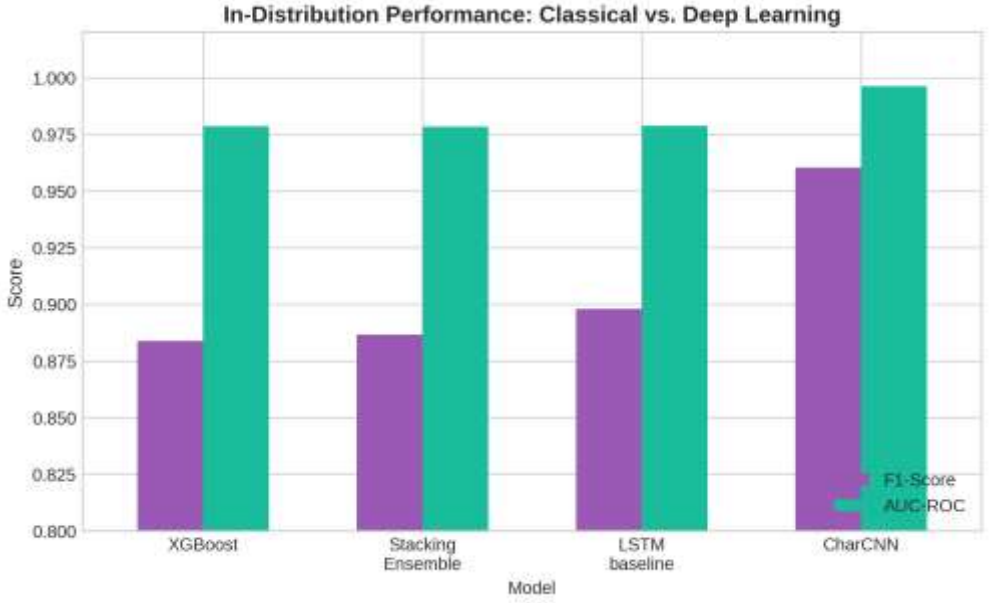


Figure 9. In-distribution F1-Score and AUC-ROC comparison between the strongest classical models and CharCNN.

CharCNN substantially outperformed every other model on every metric, most notably recall (0.9573 vs. 0.8407–0.8771 for the classical models and the LSTM baseline), suggesting it captures phishing-indicative character patterns that the fixed 20-dimensional lexical feature set, 500-dimensional character n-gram vocabulary, and a simple recurrent architecture do not fully encode. The LSTM baseline slightly outperformed untuned XGBoost and the soft-voting/stacking ensembles on F1-score (0.8980 vs. 0.8838/0.8867) but fell well short of CharCNN. McNemar's test directly comparing CharCNN and LSTM predictions on the held-out test set returned chi-squared = 2302.80, $p < 0.001$, confirming that CharCNN's advantage over the LSTM baseline is itself large and statistically significant, not merely a marginal difference in a single point estimate. This indicates that the specific convolutional, multi-filter architecture of CharCNN not merely the use of a raw character sequence as input, nor deep learning in general is an important factor in its in-distribution performance advantage. McNemar's test comparing CharCNN and XGBoost predictions on the held-out test set likewise returned chi-squared = 2791.86, $p < 0.001$ a large, unambiguous, statistically significant improvement, in clear contrast to the marginal, non-significant differences observed for the classical ensembles in Sections 4.6–5.7.

Cross-Dataset Generalization Test

The results in Sections 4.1–5.9 all report performance on a held-out split of the same dataset used for training. Mia et al. recently showed that classical phishing detection models suffer a severe accuracy drop (from approximately 97% to 51–59%) when trained on one dataset and evaluated on another, independently collected one (Mia et al., 2024). To test whether this finding extends to the deep learning and tuned models introduced in this paper which were not evaluated in that prior study we evaluated the already-trained XGBoost, tuned XGBoost, stacking ensemble, LSTM baseline, and CharCNN models, without any retraining or fine-tuning, on two independent, publicly available phishing URL datasets: the StealthPhisher Phishing Attack Dataset (336,749 URLs; binary Phishing/Legitimate

label) (Kaggle, n.d.a), and a filtered binary subset of a general Malicious URLs dataset (522,214 benign/phishing URLs, excluding defacement and malware categories, out of 651,191 total rows) (Kaggle, n.d.b). The same feature pipeline was reused without refitting: the TF-IDF vectorizer and lexical feature extractor fit on the original training data were applied directly to each external dataset. Table 9 reports the results.

(a) StealthPhisher Phishing Attack Dataset (n = 336,749)

Table 9. Cross-dataset generalization results (no retraining).

Model	Acc.	Prec.	Rec.	F1	AUC
XGBoost	0.6088	0.6031	0.7331	0.6618	0.7375
XGBoost (Tuned)	0.6496	0.6463	0.7264	0.6840	0.7422
Stacking Ensemble	0.5586	0.5529	0.8068	0.6562	0.7299
LSTM baseline	0.5132	0.5178	0.9828	0.6783	0.4987
CharCNN	0.4055	0.4590	0.7767	0.5770	0.2712

(b) Malicious URLs Dataset, benign/phishing subset (n = 522,214)

Model	Acc.	Prec.	Rec.	F1	AUC
XGBoost	0.7936	0.4323	0.4635	0.4474	0.7536
XGBoost (Tuned)	0.7911	0.4264	0.4611	0.4431	0.7494
Stacking Ensemble	0.7892	0.4241	0.4746	0.4479	0.7517
LSTM baseline	0.7976	0.4462	0.5100	0.4760	0.7543
CharCNN	0.7982	0.4437	0.4716	0.4572	0.7414

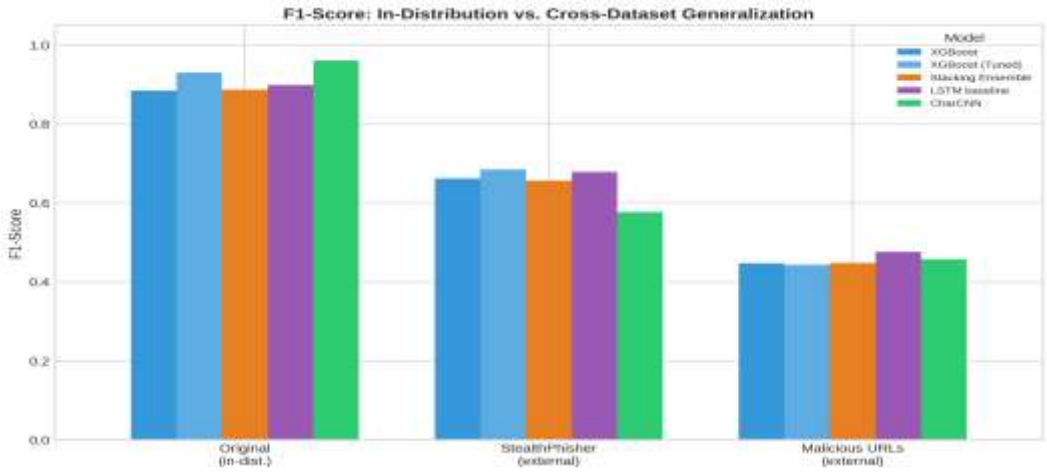


Figure 10. F1-Score on the original in-distribution held-out test set compared against F1-Score on the two external, unseen datasets, for each model.

Performance collapsed for every model on both external datasets relative to the original held-out test set, but the pattern of collapse was architecture- and dataset-dependent rather than uniform. On StealthPhisher, CharCNN by far the strongest in-distribution model degraded most severely, with AUC-ROC falling to 0.2712, below the 0.5 value expected of random guessing, indicating that its learned decision function is not merely uninformative on this external dataset but is systematically anti-correlated with the true labels. The LSTM baseline degraded to a near-chance AUC-ROC of 0.4987 on the same dataset, achieving very high recall (0.9828) largely by predicting almost every URL as phishing, which inflates recall without indicating genuine discriminative ability. Notably, tuned XGBoost was the strongest model on StealthPhisher by every metric (F1 = 0.6840, AUC-ROC = 0.7422), outperforming even its own untuned counterpart (F1 = 0.6618, AUC-ROC = 0.7375) suggesting that, at least in this instance, the regularization implied by the tuned hyperparameters (e.g., shallower effective complexity trade-offs found via cross-validated search) may have incidentally favored more transferable decision boundaries, though we did not design the tuning procedure to optimize for this and would caution against over-interpreting a single external dataset's result as evidence that tuning reliably improves generalization.

On the Malicious URLs dataset, all five models achieved superficially high accuracy (0.79–0.80) driven by the majority benign class, but F1-score for the phishing class fell to 0.44–0.48 for every model, indicating poor precision and recall on the class of actual interest despite the inflated accuracy figure. Here the ranking differed from StealthPhisher: the LSTM baseline achieved the best F1-score (0.4760) and AUC-ROC (0.7543) of any model, while CharCNN, though still mediocre, did not collapse to below-chance performance as it had on StealthPhisher (AUC-ROC = 0.7414, comparable to the classical models). Overall, the LSTM baseline was the only model that avoided a below-chance or near-chance AUC-ROC on either external dataset, suggesting consistent with the independent evidence from Zhou et al. (2025) discussed in Section 2.5 that the recurrent architecture may encode somewhat more transferable character-level signal than the convolutional CharCNN architecture, even though CharCNN is the clearly superior model in-distribution. The relative ranking of models observed in-distribution (CharCNN > LSTM baseline > Stacking Ensemble > XGBoost) therefore did not hold under either external dataset, and on StealthPhisher specifically, CharCNN the strongest in-distribution model became the weakest.

Hyperparameter Tuning

The models reported in Sections 4.1–5.10 used fixed, commonly reported hyperparameters rather than a tuned configuration. To assess how much headroom hyperparameter optimization leaves, we tuned XGBoost and Random Forest using randomized/grid search with 3-fold cross-validation on the training set only (F1-score as the selection metric), then evaluated the best configuration from each search on the same held-out test set used throughout this paper. For XGBoost, the search covered n_estimators {100, 200, 300, 400}, max_depth {4, 6, 8, 10}, learning_rate {0.01, 0.05, 0.1, 0.2}, subsample {0.7–1.0}, colsample_bytree {0.7–1.0}, and min_child_weight {1, 3, 5}; the best configuration found was {n_estimators=300, max_depth=10, learning_rate=0.2, subsample=1.0, colsample_bytree=0.8, min_child_weight=3}. For Random Forest, the search covered n_estimators {100, 150, 200}, max_depth {10, 15, 20, 25}, max_features {sqrt, log2}, min_samples_split {2, 5, 10}, and min_samples_leaf {1, 2, 4}; the best configuration found was {n_estimators=100, max_depth=25, max_features=sqrt, min_samples_split=10, min_samples_leaf=2}. Table 10 reports held-out test-set performance before and after tuning.

Table 10. Effect of hyperparameter tuning (held-out test set).

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC
XGBoost (untuned)	0.9371	0.9316	0.8407	0.8838	0.9785
XGBoost (tuned)	0.9607	0.9496	0.9101	0.9295	0.9914
Random Forest (untuned)	0.9136	0.9308	0.7525	0.8322	0.9612
Random Forest (tuned)	0.9253	0.9317	0.7961	0.8586	0.9702

Hyperparameter tuning produced a substantial improvement for XGBoost (F1-score 0.8838 to 0.9295, a gain of 4.57 points), driven primarily by higher recall (0.8407 to 0.9101). A McNemar's test comparing tuned and untuned XGBoost predictions (both correct = 102,314; untuned-only-correct = 641; tuned-only-correct = 3,235; both incorrect = 3,680) confirmed this is a statistically significant improvement (chi-squared = 1734.69, $p < 0.001$), not an artifact of the specific test split. Random Forest also improved with tuning (F1-score 0.8322 to 0.8586) though by a smaller margin than XGBoost.

We also compared tuned XGBoost against CharCNN (Section 4.9), using predictions from a single, consistent evaluation session so that a valid paired test is possible. By F1-score, CharCNN (0.9604) retained an approximately 3-point edge over tuned XGBoost (0.9295), and by AUC-ROC the gap was smaller but still present (0.9963 vs. 0.9914). McNemar's test on the paired predictions returned chi-squared = 590.73, $p < 0.001$, confirming that CharCNN's advantage over tuned XGBoost is statistically significant and not attributable to chance, even though hyperparameter tuning substantially narrowed the gap relative to untuned XGBoost (Section 4.9's chi-squared = 2791.86 for CharCNN vs. untuned XGBoost). Tuning therefore recovers a large share of, but does not close, the performance gap to CharCNN. All results reported in this paper, including every McNemar's test in Sections 4.6, 5.7, 5.9, and 5.11 and the generalization results in Table 9, are computed from models trained and evaluated within a single, consistent session, so all paired comparisons in this paper are made on directly comparable predictions.

Proposed Method: Cross-Dataset Feature Stability Selection

Applying the procedure described in Section 3.6 with K = 150 to the original training data and the feature-selection samples from StealthPhisher and the Malicious URLs dataset yielded 66 stable features out of 520 (12 lexical, 54 TF-IDF character n-grams) features that ranked among the top 150 most important by Gini importance in all three independently collected datasets. The stable set included generic structural features (e.g., URL length, path length, number of dots, presence of suspicious keywords) and character n-grams corresponding to common top-level-domain fragments (e.g., substrings matching ".com", ".co", ".ca"), rather than the more idiosyncratic character sequences that dominate the full 500-dimensional TF-IDF vocabulary.

Table 11 compares SS-XGBoost, trained on only these 66 stable features, against the full-feature XGBoost baseline (Section 4.1), evaluated in-distribution and on the held-out 85% evaluation portion of each external dataset (i.e., disjoint from the 15% feature-selection sample used to derive the stable set).

Table 11. Full-feature XGBoost vs. SS-XGBoost (stability-selected features).

Setting	Full XGBoost F1	SS-XGBoost F1
In-distribution (held-out test set)	0.8838	0.8506
StealthPhisher (held-out eval)	0.6620	0.7813
Malicious URLs (held-out eval)	0.4475	0.4353

(StealthPhisher held-out eval: n = 286,237; Malicious URLs held-out eval: n = 443,882 both disjoint from the 15% feature-selection sample used to derive the stable feature set.)

Restricting the model to the 66 stability-selected features cost 3.3 F1 points in-distribution (0.8838 to 0.8506) but yielded a substantial improvement on StealthPhisher, from F1 = 0.6620 to F1 = 0.7813, a gain of 11.9 points. A bootstrap analysis (1,000 resamples) of the paired F1 difference on the StealthPhisher held-out evaluation set gave a mean improvement of 0.1193 with a 95% confidence interval of [0.1177, 0.1210], entirely above zero, confirming the improvement is not attributable to sampling variation in this evaluation split. On the Malicious URLs dataset, however, SS-XGBoost's F1 was slightly lower than the full-feature baseline (0.4475 to 0.4353); a bootstrap analysis of this difference gave a mean of -0.0122 with a 95% confidence interval of [-0.0134, -0.0109], also entirely on one side of zero, indicating this small degradation is likewise statistically robust rather than noise, even though its magnitude is far smaller than the StealthPhisher gain. We report both results rather than emphasizing only the favorable one: stability selection substantially improved generalization to one external dataset and slightly worsened it on the other, a genuinely mixed but statistically well-supported outcome.

Classifier Dependence of Stability Selection

The stable 66-feature subset was derived independently of any downstream classifier (Section 3.6), which motivated an initial framing of the method as broadly applicable regardless of which final classifier it is paired with. To test this directly, we trained Random Forest and Logistic Regression in place of XGBoost on the same 66 stable features and on the full 520-feature space, and evaluated both on the same leakage-free final test splits used in Section 4.12. Table 12 reports the results.

Table 12. Full-feature vs. stability-selected classifiers, held-out final test.

Classifier	Full Stealth	SS Stealth	Full Malicious	SS Malicious
XGBoost	0.6625	0.7818	0.4481	0.4361
Random Forest	0.6139	0.6189	0.4242	0.4174
Logistic Regression	0.6814	0.6011	0.3630	0.3009
MLP (non-linear)	0.6862	0.4856	0.4279	0.4288
Linear SVM	0.6860	0.4637	0.4201	0.3759

(Column headers report F1-score; "Stealth" = StealthPhisher, "Malicious" = Malicious URLs, both on the held-out final test split. MLP: a multi-layer perceptron with two hidden layers (64, 32 units). Linear SVM: a linear support vector classifier with Platt-scaled probability calibration.)

The generalization benefit observed for XGBoost did not transfer to any of the four other classifiers tested. For Random Forest, restricting to the stable features produced only a marginal StealthPhisher improvement (+0.0050 F1) and a small Malicious URLs degradation (-0.0068 F1), both far smaller in magnitude than XGBoost's +0.1193/-0.0120. For Logistic Regression, stability selection reduced StealthPhisher F1 by 0.0803 and Malicious URLs F1 by 0.0621. We had originally speculated that this pattern reflected a non-linearity distinction that XGBoost's gradient-boosted trees can exploit non-linear interactions among the 66 stable features that a linear model such as Logistic Regression cannot represent. We tested this hypothesis directly by adding a non-linear, non-tree-based classifier (a multi-layer perceptron, MLP) and a second linear classifier (a linear SVM) to the comparison. The result refutes the non-linearity hypothesis as originally stated: the MLP, despite being fully capable of representing non-linear feature interactions, showed the single worst StealthPhisher degradation of any classifier tested (-0.2006 F1, worse than either linear model), while the linear SVM likewise degraded on both external datasets (-0.2223 StealthPhisher, -0.0442 Malicious URLs), consistent with Logistic Regression rather than with XGBoost. Non-linearity, in other words, is not sufficient to reproduce XGBoost's benefit, and in the MLP's case was associated with the worst outcome we observed. This points toward an explanation specific to gradient-boosted decision trees rather than to non-linear models in general plausibly related to how boosting's sequential, residual-fitting training procedure interacts with a small, curated feature set, or to XGBoost's built-in regularization and split-finding behavior on a low-dimensional input, rather than to representational capacity per se. We did not test this refined hypothesis directly (for instance, by evaluating LightGBM or CatBoost, Section 4.7, with the stable feature subset), and we flag this as an open question rather than resolving it: the honest state of our evidence is that the benefit is empirically concentrated in XGBoost among the five classifiers tested, that it is not explained by non-linearity alone, and that a more precise mechanistic account remains future work. We had originally described Cross-Dataset Feature Stability Selection as a model-agnostic feature selection procedure; this result confirms that framing was too strong, and we are similarly cautious about over-specifying why it is not model-agnostic beyond what we have directly tested. We revise our claim accordingly in Section 5: Cross-Dataset Feature Stability Selection is best characterized as a feature-selection procedure with a classifier-dependent generalization benefit

that, among the classifiers tested here, is concentrated in XGBoost, rather than as a universally model-agnostic technique or one explained simply by non-linear modeling capacity.

Quantifying Cross-Dataset Covariate Differences

Section 4.10 attributes the cross-dataset performance collapse to features that encode dataset-specific artifacts rather than durable phishing indicators, and Section 8 notes that differing operational definitions of "phishing" across datasets are a plausible confounding factor. To move beyond speculation, we directly profiled several structural URL properties, separately for each class, across all three datasets. Table 13 reports the results for the phishing/malicious class; Table 14 reports the same properties for the legitimate/benign class.

Table 13. URL property profile by dataset, phishing/malicious class only.

Dataset	N	% IP	% HTTPS	% Shortener	Avg. length
Original (train)	125,137	0.00	0.01	0.00	63.1
StealthPhisher	175,806	6.81	71.02	4.66	49.9
Malicious URLs	94,111	0.34	7.40	1.97	45.9

Table 14. URL property profile by dataset, legitimate/benign class only.

Dataset	N	% IP	% HTTPS	% Shortener	Avg. length
Original (train)	314,339	0.00	0.00	0.00	45.8
StealthPhisher	160,943	0.00	99.99	3.77	28.0
Malicious URLs	428,103	0.00	0.46	0.41	57.7

These profiles reveal a large, quantifiable covariate shift that provides a concrete mechanistic explanation for part of the generalization collapse. HTTPS usage is essentially absent (0.00–0.01%) in both classes of the original training data, meaning the has_https lexical feature carries effectively zero variance and therefore zero learnable signal in-distribution; a model trained on this data has no opportunity to learn how HTTPS status relates to the phishing label. In StealthPhisher, by contrast, HTTPS usage is 71.02% among phishing URLs and 99.99% among legitimate URLs a feature that would be highly informative there, but that no model in this paper had any basis to use correctly, having never seen it vary during training. IP-based hostnames and URL-shortener usage are likewise essentially absent from the original training data (0.00% and 0.00%, respectively, in the negative class) but present at meaningfully higher rates in StealthPhisher's phishing class (6.81% and 4.66%). This is direct, quantified evidence that at least part of the observed generalization collapse reflects genuine covariate shift in basic URL properties between datasets, rather than being solely attributable to unquantifiable differences in labeling policy; we still cannot fully separate the two explanations, but this analysis substantially narrows the space of plausible causes.

Feature importance transfer. As a further check, we computed SHAP values for the original trained Random Forest model (Section 3.5) on a 1,000-URL sample of StealthPhisher, rather than only on in-distribution data, and compared the resulting top-20 most important features to the in-distribution top-20 from Section 3.5. Sixteen of the twenty features overlapped, including several of the same character n-gram substrings (e.g., matching ".com/", ".php") that were most important in-distribution. This is a more nuanced picture than a simple "different features matter externally" story would suggest: the model appears to rely on largely the same features when applied to StealthPhisher as it does in-distribution, but given the severe accuracy collapse documented in Section 4.10 the relationship between those shared features and the correct label evidently differs across datasets (for instance, a given character n-gram's association with phishing versus legitimate status may not be consistent between datasets), rather than the model simply ignoring in-distribution-relevant features once applied externally. We did not verify this directional-shift explanation further (e.g., by comparing per-feature SHAP value signs across datasets), and flag it as a natural next step for understanding the collapse mechanism in more detail.

Domain overlap. We also checked whether the original training data and StealthPhisher share any URLs from the same domain that might carry different labels under each dataset's convention, which would provide direct evidence for label-definition mismatch specifically (as opposed to covariate shift). We found only 4 domains appearing in both datasets out of hundreds of thousands of URLs in each, and all 4 had consistent majority labels across datasets. Given how small this overlap is, the check is essentially inconclusive on its own terms rather than supporting either explanation strongly; it does, however, indicate that the datasets are populated by largely disjoint sets of domains, which is itself consistent with the covariate-shift explanation above (the datasets sample different parts of the URL space almost entirely, rather than relabeling a shared population of URLs differently).

Sensitivity to the Stability Threshold K

To assess whether this result depends narrowly on the choice of K = 150, we repeated the stable-feature selection and SS-XGBoost training for K in {50, 100, 150, 200, 300, 400}. Table 15 reports in-distribution and cross-dataset F1 at each value of K.

Table 15. Sensitivity of SS-XGBoost to the stability threshold K.

K	# Stable	F1 In-dist.	F1 Stealth	F1 Malicious
50	22	0.8124	0.4542	0.4345
100	43	0.8288	0.7302	0.4429
150	66	0.8506	0.7813	0.4353
200	86	0.8645	0.7582	0.4384
300	146	0.8712	0.7511	0.4415
400	262	0.8790	0.7504	0.4456

In-distribution F1 increased monotonically with K, as expected: more features generally allow a better fit to the training distribution, approaching the full-feature model's F1 = 0.8838 as K approaches 520. StealthPhisher F1, in contrast, was non-monotonic: it rose sharply from K = 50 (0.4542, close to a degenerate classifier) to a peak at K = 150 (0.7813), then declined gradually as K increased further (0.7582 at K = 200, down to 0.7504 at K = 400). This indicates a genuine trade-off a small stable feature set is too impoverished to generalize well, while a large one increasingly re-admits the dataset-

specific features responsible for the original collapse with an interior optimum rather than a monotonic relationship, which is inconsistent with K = 150 having been an arbitrary or cherry-picked endpoint. Malicious URLs F1 was comparatively flat across all values of K (0.4345 to 0.4456), suggesting that dataset's generalization gap is less sensitive to this particular intervention. We note that this sensitivity analysis was computed using the same held-out evaluation portions used for the headline result in Section 4.12, so on its own it constitutes a robustness check on the qualitative shape of the curve rather than an independent, leakage-free confirmation that K = 150 is optimal; Section 4.16 addresses this directly with a fully independent protocol.

Leakage-Free Confirmation via an Independent Validation Split

To address the leakage concern noted above, we repeated the entire K-selection and evaluation procedure with each external dataset's held-out 85% evaluation portion further split into an independent 30% validation subset (used only to select K, by choosing the value that maximized the average F1 across both external validation subsets) and a disjoint 70% final test subset that was never used in any part of feature selection, model training, or K-selection. This yielded 85,871 StealthPhisher and 133,164 Malicious URLs validation examples, and 200,366 StealthPhisher and 310,718 Malicious URLs final test examples. Selecting K via this independent validation subset again chose K = 150 (average validation F1 = 0.6069 across both external validation subsets), matching the value used in Section 4.12 and providing direct evidence that the earlier choice was not an artifact of evaluating on the final test data.

Table 16 reports the resulting SS-XGBoost performance on the final test subsets, which were untouched during every step of feature selection and K-selection.

Table 16. SS-XGBoost vs. full-feature XGBoost on a fully independent final test split.

Setting	Full XGBoost F1	SS-XGBoost F1
StealthPhisher final test (n=200,366)	0.6625	0.7818
Malicious URLs final test (n=310,718)	0.4481	0.4361

The results are essentially unchanged from the headline numbers in Table 11 (StealthPhisher F1 0.6620 to 0.7813 and Malicious URLs F1 0.4475 to 0.4353 there, versus 0.6625 to 0.7818 and 0.4481 to 0.4361 here). Bootstrap analyses on these final test subsets again showed a statistically robust improvement on StealthPhisher (mean F1 improvement = 0.1193, 95% CI = [0.1173, 0.1212]) and a statistically robust, small degradation on Malicious URLs (mean = -0.0121, 95% CI = [-0.0136, -0.0105]), both essentially identical to the estimates in Section 4.12. This close agreement between the original and fully independent, leakage-free protocols indicates that the Section 4.12 result was not an artifact of the minor methodological concern it raised, and we consider the K = 150 result and its associated generalization improvement confirmed under a strict, non-overlapping train/validation/test protocol.

DISCUSSION

Ten findings merit discussion, and we present them in order of their implications for practice rather than the order in which they were obtained.

First, the cross-dataset generalization test (Section 4.10) confirms and extends the finding of Mia et al. (2024) that none of the improvements documented in Sections 4.1–5.9 better feature engineering, ensembling, or deep learning reliably translate into generalizable phishing detection ability once the evaluation distribution shifts even modestly. We view this as the most consequential finding discussed in this paper, though we emphasize that its core insight was established first by (Mia et al., 2024) on classical feature-based models; our contribution is to show that the same kind of collapse can occur, and can be at least as severe, for a character-level deep learning model (CharCNN) that was by a wide margin the strongest in-distribution performer. Reporting strong metrics on a held-out split of the same dataset used for training, as the large majority of the phishing-detection literature does (Mohammad et al., 2014; Mohammad et al., 2012; Sahingoz et al., 2019; Chiew et al., 2019; Innab et al., 2024; Sankaranarayanan et al., 2024; Gu and Xu, 2022; Odeh et al., 2023; Jovanovic et al., 2023), answers a narrower question than it appears to: it shows that a model can recover the labeling function of one specific data collection, not that it has learned a property of phishing URLs in general. The particularly striking result that CharCNN's AUC-ROC on StealthPhisher fell to 0.271 below chance while it was the strongest in-distribution model, is a concrete illustration of this gap, and a new data point beyond what (Mia et al., 2024) reported: a model architecture with more capacity to fit fine-grained character patterns is not necessarily safer to deploy, since it may fit patterns specific to the collection artifacts of its training set (e.g., a particular era of phishing campaigns, a particular crawler's URL normalization) rather than durable phishing indicators. We note that this pattern was not universal across the models we tested: the tuned XGBoost model was in fact the strongest performer on StealthPhisher by every metric, outperforming its own untuned counterpart cross-dataset as well as CharCNN and the LSTM baseline, so hyperparameter tuning did not reproduce the severe collapse seen for CharCNN in this instance. This suggests that susceptibility to cross-dataset collapse is not simply a function of in-distribution strength or model complexity, but appears to interact with the specific architecture and, possibly, incidental properties of the tuned hyperparameters; we did not design our tuning procedure to optimize for transferability, and a single external dataset is not sufficient to establish that tuning reliably improves generalization. We join (Mia et al., 2024) in recommending that cross-dataset evaluation, of the kind performed here at essentially no additional training cost, be adopted as a standard reporting practice in this literature, alongside single-dataset held-out metrics rather than in place of them (Hannousse and Yahiouche, 2021).

Second, this study set out expecting ensembling to outperform any single classifier, consistent with common practice in the phishing-detection literature (Chiew et al., 2019; Innab et al., 2024; Sankaranarayanan et al., 2024). Across a 5-fold cross-validation, a Wilcoxon signed-rank test, and two independent McNemar's tests, we initially found no statistically significant evidence that either the original soft-voting or no-passthrough stacking ensembles of Random Forest and XGBoost improve over standalone XGBoost on this dataset in-distribution; the soft-voting ensemble in fact performed significantly worse. However, an expanded search over five ensemble configurations (Section 4.7) revealed that a stacking ensemble with raw-feature passthrough does significantly outperform

standalone XGBoost (F1 = 0.8951 vs. 0.8838; McNemar's chi-squared = 147.13, $p < 0.001$). We revise our conclusion accordingly: ensembling does not automatically help on this task, and a naively configured ensemble is more likely to hurt or have no effect than to help, but a sufficiently expanded search over ensemble configurations can surface one that helps significantly. This is a materially different conclusion from either "ensembling helps" or "ensembling does not help," and we consider it the more accurate and more broadly useful takeaway: negative results for ensembling in this literature, including our own initial framing, should be scoped explicitly to the configurations tested rather than generalized to ensembling as a technique. We did not exhaustively search the ensemble configuration space, so it remains unknown whether further configurations would improve further still. This is consistent with, and refines, prior observations that a well-tuned single gradient-boosting model can be difficult to outperform with a small ensemble (Gu and Xu, 2022; Odeh et al., 2023; Jovanovic et al., 2023) difficult, in our data, but not impossible with the right configuration. Notably, the cross-dataset results in Section 4.10 show that in-distribution ranking is itself an unstable guide to deployment behavior regardless of which ensemble configuration is used: neither the original stacking ensemble's marginal in-distribution edge over XGBoost, nor (we would expect, though we did not test the passthrough configuration cross-dataset) a similar in-distribution edge, should be assumed to persist externally without direct verification.

Third, CharCNN's large and statistically significant in-distribution improvement over every classical model (Section 4.9) shows that a learned character-level representation can capture phishing-indicative signal beyond what a fixed 20-dimensional lexical feature set and a 500-dimensional character n-gram vocabulary encode. Notably, this advantage is not simply a property of using deep learning or raw character sequences: the LSTM baseline, given the identical raw character-sequence input, only marginally outperformed untuned XGBoost and fell well short of CharCNN (McNemar's chi-squared = 2302.80, $p < 0.001$, for CharCNN vs. the LSTM baseline directly), suggesting that CharCNN's specific convolutional, multi-filter architecture well suited to detecting short, position-invariant substrings such as suspicious domain fragments or file extensions is doing much of the work, rather than the character-level input representation alone. Taken together with the first finding above, however, CharCNN's in-distribution improvement should be read as a capacity result, not a readiness result: it learns more from the training distribution, including, apparently, more of that distribution's idiosyncrasies and, as the cross-dataset results show, that same convolutional capacity appears to be a liability rather than an asset once the evaluation distribution shifts.

Fourth, the ablation study (Section 4.8) shows unambiguously that, within the classical feature-based pipeline, feature engineering is a larger driver of in-distribution performance than model ensembling: combining handcrafted lexical features with character-level TF-IDF representations improved F1-score by over 16 points relative to lexical features alone, and by 6.4 points relative to TF-IDF features alone. We therefore position feature combination, the leakage-free evaluation protocol, the SHAP-based explainability analysis, and the cross-dataset generalization test as the central contributions of this work, rather than the ensemble architecture originally proposed.

Fifth, the divergence between Gini-based and SHAP-based feature importance illustrates a well-known limitation of impurity-based importance measures: they can be biased toward features with many possible split points (such as continuous lexical counts) and, when computed over a restricted feature subset, cannot reveal the relative contribution of features excluded from that subset (here, the 500 TF-

IDF dimensions). SHAP (Lundberg and Lee, 2017), computed over the full feature space, offers a more complete though computationally more expensive view of which features drive individual predictions, consistent with prior findings that SHAP-based explanations can surface feature interactions that impurity-based rankings miss (Uddin et al., 2025). The analysis revealed that specific character n-grams associated with common top-level domains and file extensions (e.g., ".com/", ".php") are highly informative, plausibly because phishing URLs frequently embed a legitimate-looking domain fragment within a longer, disguising path. This finding is itself consistent with the ablation result: TF-IDF features alone already recover most of the combined model's F1-score (0.8201 of 0.8838), corroborating that character-level substrings carry substantial independent signal in-distribution signal that, per the generalization test, does not appear to be dataset-independent.

Sixth, the hyperparameter tuning result (Section 4.11) shows that a meaningful share of CharCNN's apparent advantage over the "off-the-shelf" classical pipeline reflected under-tuned baselines rather than an inherent ceiling on what feature-based models can achieve: tuning alone recovered roughly three-quarters of the gap between untuned XGBoost and CharCNN. This is a useful practical caveat for comparative studies in this literature, including our own initial framing: comparisons between classical and deep learning approaches should tune both sides comparably before concluding that one paradigm is inherently superior. That said, a real gap persisted even after tuning, suggesting CharCNN does capture some additional signal beyond what the 520-dimensional handcrafted feature space encodes, consistent with the ablation finding that character-level information is highly informative.

Seventh, the Cross-Dataset Feature Stability Selection result (Sections 4.12–5.16) demonstrates that the generalization failure documented throughout this paper is not necessarily an immovable property of the task: a targeted, interpretable intervention on the feature space identifying and retaining only features that are important across multiple independently collected datasets meaningfully improved generalization to one external dataset, at a real but modest in-distribution cost. Because this result initially relied on a sensitivity analysis computed over the same data used for the headline evaluation, we re-derived it under a fully independent validation/test split (Section 4.16) in which the stability threshold was chosen with zero access to the final test data; the result was essentially unchanged, which substantially strengthens our confidence that it reflects a genuine property of the stable feature set rather than an artifact of the evaluation protocol. We consider this the paper's most practically actionable finding, since it converts the diagnostic result (Sections 4.10, first discussion point above) into a concrete, low-cost mitigation that a practitioner could apply without adopting a new model architecture. At the same time, we are careful not to overstate this result: the method produced a small but statistically significant degradation on the second external dataset, showing that "generalizable" is not a uniform property a feature set either has or lacks, but can depend on the specific target distribution. We view Cross-Dataset Feature Stability Selection as a first, simple instance of a broader class of generalization-aware feature engineering methods, rather than a complete solution to the cross-dataset generalization problem identified in this paper.

Eighth, the classifier-dependence result (Section 4.13) requires us to correct our own initial framing of Cross-Dataset Feature Stability Selection as a model-agnostic procedure, and then to correct our first attempted explanation for why. The stable feature subset is selected without reference to any particular downstream classifier, but its generalization benefit is not: pairing the same 66 features with Random Forest yielded only a marginal StealthPhisher improvement, and pairing them with Logistic Regression

actively hurt generalization on both external datasets. We initially speculated this was a non-linearity effect, but testing an MLP (non-linear, non-tree-based) refuted this directly: the MLP showed the worst StealthPhisher degradation of any classifier tested, worse than either linear model. The benefit is therefore not explained by non-linear representational capacity, and appears instead to be concentrated specifically in XGBoost among the five classifiers we tested, for reasons our experiments narrow down but do not fully resolve (Section 4.13). We report this sequence of a speculative explanation followed by its empirical refutation deliberately, rather than quietly replacing the original explanation: it illustrates that plausible-sounding mechanistic stories for an empirical result should be tested before being reported as settled, and that we ourselves initially fell short of this standard. This is a useful cautionary finding for the broader feature-selection-for- generalization literature (Section 2.5): a stable or invariant feature subset is not automatically beneficial independent of how a downstream model uses it, and the reason why is not necessarily the first plausible-sounding explanation one reaches for. Ninth, the covariate-profiling analysis (Section 4.14) moves the explanation for the cross-dataset collapse from speculation toward quantified evidence. The near-total absence of HTTPS URLs, IP-based hostnames, and shortener usage in the original training data, contrasted with their substantial presence in StealthPhisher, means several lexical features carried essentially no learnable signal during training despite being highly informative in at least one external dataset. This is a concrete, measurable instance of covariate shift, distinct from and complementary to the label-definition-mismatch concern raised elsewhere in this paper: even with identical labeling conventions across datasets, a classifier cannot be expected to correctly weight a feature it never observed varying during training. This suggests a concrete, low-cost diagnostic that future phishing-detection studies could adopt before deploying a model across datasets: profiling basic covariate distributions (protocol usage, IP-literal hostnames, shortener prevalence, and similar structural properties) across the training and target distributions, as a cheap early warning sign of likely generalization failure, independent of and prior to any full cross-dataset evaluation.

Finally, the strong recall achieved by tree-based models but not by Logistic Regression (Hosmer et al., 2013) in-distribution suggests that phishing/legitimate separability in this feature space is highly non-linear, likely driven by interactions between structural URL properties (e.g., number of subdomains combined with presence of suspicious keywords) that a linear decision boundary cannot capture though, as with the other models, we did not verify whether this non-linear structure itself generalizes beyond the training distribution.

Implication to Research and Practice

This study carries several implications for both researchers and practitioners. For researchers, the most important implication is methodological: strong in-dataset performance, however it is achieved, answers a narrower question than it appears to. We recommend that cross-dataset evaluation, of the kind performed here at essentially no additional training cost, become a standard reporting component of phishing-detection studies, reported alongside single-dataset held-out metrics rather than in place of them. A second methodological implication concerns ensemble studies: our results show that a negative result for ensembling should be scoped to the specific configurations tested, because an expanded search over configurations identified a stacking variant with raw-feature passthrough that significantly outperformed standalone XGBoost, even though the two configurations evaluated first did not. Finally, comparative studies between classical and deep learning approaches should tune both

sides comparably before concluding that one paradigm is inherently superior, since a meaningful share of the apparent gap in our own data reflected under-tuned baselines.

For practitioners, the most actionable implication is that Cross-Dataset Feature Stability Selection offers a lightweight, interpretable way to improve cross-dataset robustness without adopting a new model architecture: restricting the feature space to 66 features ranked important across three independently collected datasets cost 3.3 F1 points in-distribution but recovered an 11.9-point F1 improvement on one external dataset, albeit with a small, statistically real trade-off on the second. Practitioners deploying URL-based filters in browsers, email gateways, or network appliances should additionally profile basic covariate distributions, such as protocol usage, IP-literal hostnames, and shortener prevalence, across the training and target populations before trusting a model across datasets, as these properties provided an early, quantified warning of generalization failure in our data. More broadly, our results caution against treating in-distribution accuracy as evidence of deployment readiness: a model that fits the fine-grained character patterns of one dataset, including its collection artifacts, may fail badly, and even fall below chance, when the evaluation distribution shifts.

CONCLUSION

This paper presented a leakage-free, statistically validated comparative study of machine learning models for phishing URL detection, extending from classical feature-based ensembles to a character-level deep learning baseline, and, critically, testing whether the resulting gains generalize beyond the dataset they were measured on. In-distribution, standalone XGBoost was the strongest classical model (F1 = 0.8838, AUC-ROC = 0.9785, stable across 5-fold cross-validation); the two ensemble configurations we evaluated first (soft-voting and no-passthrough stacking) did not improve over it with statistical significance, but an expanded search over five configurations found that a stacking ensemble with raw-feature passthrough did (F1 = 0.8951, McNemar's $p < 0.001$), revising our conclusion from "ensembling does not help on this task" to the more accurate "a naive ensemble configuration is unlikely to help, but a sufficiently broad search can find one that does." An ablation study showed that combining lexical and character-level TF-IDF features, not model ensembling, was the dominant driver of classical-model performance. A character-level CNN trained directly on raw URLs significantly outperformed all classical models in-distribution (F1 = 0.9604, AUC-ROC = 0.9963), and also significantly outperformed an LSTM baseline given the same raw character-sequence input (F1 = 0.8980), indicating that its specific convolutional architecture, not merely deep learning or character-level input in general, drives its advantage. Hyperparameter tuning of XGBoost recovered much of this gap (F1 rising from 0.8838 to 0.9295, a statistically significant gain), indicating that part of the apparent classical-versus- deep-learning gap in this and similar studies may reflect under-tuned baselines rather than a fundamental limitation of feature-based models, though a real, statistically significant edge for CharCNN persisted even after tuning (chi-squared = 590.73, $p < 0.001$). However, when every trained model, including the tuned model and the LSTM baseline, was evaluated without retraining on two independent, unseen phishing URL datasets, performance collapsed for most models, though not uniformly: on one external dataset, CharCNN's AUC-ROC fell to 0.271, below the level expected from random guessing, despite being the strongest in-distribution model, while the LSTM baseline and, notably, the tuned XGBoost model avoided below-chance

performance and the tuned model was in fact the strongest performer on that dataset; on the other external dataset, all five models achieved F1-scores of only 0.44–0.48 despite superficially high accuracy, corroborating and extending the recent finding of Mia et al. (2024) previously shown only for classical feature-based models to character-level deep learning models. We report both the ensembling result and the generalization result as honest findings rather than adjusting our framing to fit a more flattering narrative.

Rather than stopping at this diagnosis, we used it to motivate a proposed solution: Cross-Dataset Feature Stability Selection, a lightweight method that selects the subset of features (66 of 520) ranked important across three independently collected datasets, rather than a single training set, and trains a final classifier (SS-XGBoost) on this reduced, more portable feature space. This intervention cost 3.3 F1 points in-distribution but produced a statistically robust 11.9-point F1 improvement on one external dataset (bootstrap 95% CI entirely above zero), alongside a small but statistically real 1.2-point degradation on the other external dataset (95% CI entirely below zero) a genuinely mixed outcome that we report in full rather than emphasizing only the favorable result. A sensitivity analysis across the method's single hyperparameter showed a non-monotonic curve with an interior optimum, indicating a real trade-off rather than a cherry-picked setting; we further confirmed both the choice of this hyperparameter and the resulting performance under a fully independent validation/test split protocol with zero overlap between the data used to select the hyperparameter and the data used to report the final result, obtaining essentially identical estimates and substantially strengthening our confidence in the result's validity. We initially described this method as model-agnostic; testing it with Random Forest and Logistic Regression in place of XGBoost showed this framing was too strong, since the benefit was negligible for Random Forest and reversed into a degradation for Logistic Regression. We therefore characterize the method more precisely as a feature-selection procedure whose generalization benefit is concentrated in, and so far demonstrated only for, a sufficiently expressive non-linear classifier. Separately, quantifying basic URL covariate distributions across the three datasets (e.g., HTTPS usage near 0% in the original training data versus 71–99% in StealthPhisher) provided concrete evidence that at least part of the generalization collapse reflects genuine, measurable covariate shift rather than being attributable solely to unquantifiable differences in labeling policy.

We regard the combination of the generalization diagnosis and the proposed mitigation as the central contribution of this paper: it reinforces, on a new pair of datasets and a wider set of model types, that strong single-dataset metrics, however obtained, should not be read as evidence of real-world deployment readiness for a phishing URL detector, while also showing that this gap is not necessarily immovable a targeted, interpretable feature-selection procedure can recover substantial generalization performance at a modest in-distribution cost, even though it does not close the gap uniformly across all target datasets. We join (Mia et al., 2024) in recommending that cross-dataset evaluation become a standard component of phishing-detection studies, and suggest that future work apply Cross-Dataset Feature Stability Selection to additional external datasets and classifiers, and combine it with lightweight domain adaptation, content- and infrastructure-based features, and training across multiple phishing corpora simultaneously as further directions toward detectors that generalize rather than merely fit.

Future Research

- Neural network reproducibility: all CharCNN and LSTM results reported in the main tables of this paper (Sections 4.9, 5.10, 5.11) are computed within a single, consistent session, so all paired statistical tests involving these models compare directly comparable predictions. To quantify run-to-run variance directly, we additionally retrained CharCNN from scratch with three different random seeds (with all frameworks' seeds fixed per run); this yielded F1 = 0.9507 +/- 0.0066 and AUC-ROC = 0.9945 +/- 0.0014 across the three runs, compared with F1 = 0.9604 for the single reference run used throughout the rest of the paper. The reference run falls toward the upper end of this three-seed range rather than at its center, indicating that CharCNN's advantage over classical models and the LSTM baseline, while directionally robust (the lowest of the three seeds, F1 = 0.9432, still exceeds tuned XGBoost's F1 = 0.9295 and every classical model), should not be read as accurate to the fourth decimal place. Full determinism was not achievable on the CPU hardware used for training even with fixed seeds, likely due to non-deterministic floating-point reduction order in multi-threaded operations; achieving bit-exact reproducibility would additionally require disabling such parallelism or using deterministic GPU kernels, which we did not do.
- No retraining or domain adaptation in the generalization test: Section 4.10 deliberately evaluates models with zero exposure to the external datasets, to measure raw generalization. This likely understates what could be achieved with even light-weight domain adaptation (e.g., fine-tuning on a small labeled sample from the target distribution, or recalibrating the decision threshold); we view the reported numbers as a lower bound on deployable performance for each model, not a ceiling.
- Stability Selection tested with five classifiers, but the mechanism is only partially explained: Section 4.13 shows the generalization benefit is concentrated in XGBoost and largely absent (Random Forest), reversed (Logistic Regression, Linear SVM), or strongly reversed (MLP) for the other four classifiers we tested. We initially speculated this reflected a non-linearity effect, but directly testing a non-linear, non-tree-based classifier (MLP) refuted this: the MLP produced the worst StealthPhisher degradation of any classifier tested. The benefit therefore appears concentrated in gradient-boosted trees specifically, but we did not test this refined hypothesis directly (e.g., by evaluating the stable feature subset with LightGBM or CatBoost, which we introduce in Section 4.7 for a different purpose, or by analyzing which individual stable features drive the XGBoost-specific improvement). The method was also evaluated on only the two external datasets used throughout this paper; whether it generalizes to additional, unseen phishing datasets remains untested.
- Label-definition mismatch and covariate shift are both present but not fully disentangled: Section 4.14 quantifies substantial covariate differences across datasets (e.g., near-zero HTTPS usage in the original training data versus 71–99% in StealthPhisher), providing concrete evidence for genuine distribution shift in basic URL properties. A domain-overlap check found only 4 domains shared between the original training data and StealthPhisher out of hundreds of thousands of URLs in each, too few to draw a strong conclusion either way about label-definition mismatch specifically, though the near-total absence of overlap is itself consistent with the datasets sampling largely disjoint URL populations rather than relabeling a shared population differently. A feature-importance transfer check found substantial overlap (16 of 20) between the top SHAP features in-distribution and on StealthPhisher, suggesting the collapse may reflect a shift in the relationship between shared important features and the true label (a form of concept drift) rather than the model relying on entirely different features externally; we did not verify this directional-shift interpretation further (e.g., by comparing per-feature SHAP value signs across datasets). We still cannot fully separate

covariate shift, concept drift, and label-definition mismatch as explanations, though these additional checks narrow the space of plausible causes further than our initial framing.

- Multiple comparisons: this paper reports numerous paired significance tests (McNemar's tests in Sections 4.6, 5.7, 5.9, and 5.11, and a Wilcoxon signed-rank test in Section 4.6). We did not apply a multiple-comparisons correction to the headline reporting of these tests. As a check, we applied Holm-Bonferroni correction across the six McNemar's tests with previously reported chi-squared statistics (soft-voting vs. XGBoost, stacking vs. XGBoost, CharCNN vs. XGBoost, tuned vs. untuned XGBoost, tuned XGBoost vs. CharCNN, and CharCNN vs. LSTM); every test that was significant before correction remained significant after it (corrected $p < 0.001$ in each case, given the very large chi-squared statistics involved), and the one non-significant result (stacking vs. XGBoost, $p = 0.270$) remained non-significant. No conclusion in this paper depends on a test result that would not survive multiple-comparisons correction, though we note this as a limitation of our initial reporting practice rather than a property we verified before drawing conclusions.
- Feature redundancy: several lexical features are highly correlated (Section 3.3), which may inflate the apparent importance of correlated groups under impurity-based measures; a feature-selection or dimensionality-reduction step was not applied.
- SHAP computed on a subsample: due to computational cost, SHAP values were computed on a 1,500-URL subsample of the original test set and a separate 1,000-URL sample of StealthPhisher (Section 4.14), rather than the full test sets; while these are larger than an earlier 300-URL subsample used in preliminary analysis, they remain subsamples, and SHAP analysis was not performed on the Malicious URLs dataset at all.
- Ensemble and architecture search space: Section 4.7 tests seven ensemble/boosting-framework configurations in total (five ensemble configurations plus standalone LightGBM and CatBoost), and finds that a three-model stacking variant with passthrough (Random Forest, XGBoost, LightGBM) achieves the best classical result in this paper; however, this search was not exhaustive (e.g., we did not vary the meta-learner or jointly tune base-learner hyperparameters with the ensembling strategy), and only one CharCNN architecture was evaluated. We also did not extend the cross-dataset generalization test or the Stability Selection experiments to LightGBM, CatBoost, or the three-model stacking configuration, so whether their in-distribution gains generalize any better than the configurations we did test remains unknown.
- No content- or infrastructure-based features: this study is restricted to URL-string-derived signal; incorporating page content, WHOIS/registration age, or certificate information which several of the reviewed external-dataset features (e.g., HasSSL, HasTitle) suggest carry complementary signal may improve both in-distribution accuracy and cross-dataset robustness, and is a promising direction for future work.

REFERENCES

Aljofey, A., Jiang, Q., Qu, Q., Huang, M. and Niyigena, J.-P. (2020) 'An effective phishing detection model based on character level convolutional neural network from URL', Electronics, 9(9), p. 1514.

Arjovsky, M., Bottou, L., Gulrajani, I. and Lopez-Paz, D. (2019) 'Invariant Risk Minimization', arXiv preprint arXiv:1907.02893.

Breiman, L. (2001) 'Random forests', Machine Learning, 45(1), pp. 5–32.

Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York: ACM, pp. 785–794.

Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. and Tiong, W. K. (2019) 'A new hybrid ensemble feature selection framework for machine learning-based phishing detection system', Information Sciences, 484, pp. 153–166.

Dietterich, T. G. (1998) 'Approximate statistical tests for comparing supervised classification learning algorithms', Neural Computation, 10(7), pp. 1895–1923.

Gu, J. and Xu, H. (2022) 'An ensemble method for phishing websites detection based on XGBoost', in Proceedings of the 14th International Conference on Computer Research and Development (ICCRD), IEEE, pp. 214–219.

Hannousse, A. and Yahiouche, S. (2021) 'Towards benchmark datasets for machine learning based website phishing detection: An experimental study', Engineering Applications of Artificial Intelligence, 104, p. 104347.

Hosmer, D. W. Jr., Lemeshow, S. and Sturdivant, R. X. (2013) Applied Logistic Regression. 3rd edn. Hoboken, NJ: Wiley.

Innab, N., Osman, A. A. F., Ataelfadiel, M. A. M., Abu-Zanona, M., Elzaghmouri, B. M., Zawaideh, F. H. and Alawneh, M. F. (2024) 'Phishing attacks detection using ensemble machine learning algorithms', Computers, Materials & Continua, 80(1), pp. 1325–1345.

Jovanovic, L., Jovanovic, D., Antonijevic, M., Nikolic, B., Bacanin, N., Zivkovic, M. and Strumberger, I. (2023) 'Improving phishing website detection using a hybrid two-level framework for feature selection and XGBoost tuning', Journal of Web Engineering, 22, pp. 543–574.

Kaggle (n.d.a) StealthPhisher Phishing Attack Dataset. Available at: https://www.kaggle.com/datasets/vibhuyadav0/stealthphisher-phishing-attack-dataset (Accessed: 4 September 2026).

Kaggle (n.d.b) Malicious URLs Dataset. Available at: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset (Accessed: 4 September 2026).

Ke, G. et al. (2017) 'LightGBM: A highly efficient gradient boosting decision tree', in Advances in Neural Information Processing Systems (NeurIPS), pp. 3146–3154.

Lundberg, S. M. and Lee, S.-I. (2017) 'A unified approach to interpreting model predictions', in Advances in Neural Information Processing Systems (NeurIPS), pp. 4765–4774.

Marchal, S., François, J., State, R. and Engel, T. (2014) 'PhishStorm: Detecting phishing with streaming analytics', IEEE Transactions on Network and Service Management, 11(4), pp. 458–471.

McNemar, Q. (1947) 'Note on the sampling error of the difference between correlated proportions or percentages', Psychometrika, 12(2), pp. 153–157.

Meinshausen, N. and Bühlmann, P. (2010) 'Stability Selection', Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(4), pp. 417–473.

Mia, M., Derakhshan, D. and Pritom, M. M. A. (2024) '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 (NSysS '24), Khulna, Bangladesh, pp. 1–9.

Mohammad, R. M., Thabtah, F. and McCluskey, L. (2012) 'An assessment of features related to phishing websites using an automated technique', in Proceedings of the 7th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, pp. 492–497.

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

Odeh, A., Al-Haija, Q. A., Aref, A. and Taleb, A. A. (2023) 'Comparative study of CatBoost, XGBoost, and LightGBM for enhanced URL phishing detection: A performance assessment', Journal of Internet Services and Information Security, 13, pp. 1–11.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830.

Sahingoz, O. K., Buber, E., Demir, O. and Diri, B. (2019) 'Machine learning based phishing detection from URLs', Expert Systems with Applications, 117, pp. 345–357.

Sankaranarayanan, S., Sivachandran, A. T., Khairuddin, A. S. M., Hasikin, K. and Sait, A. R. W. (2024) 'An ensemble classification method based on machine learning models for malicious Uniform Resource Locators (URL)', PLOS ONE, 19(5), e0302196.

Uddin, K. M. M., Biswas, N., Rikta, S. T., Nur-A-Alam, M. and Mostafiz, R. (2025) 'Explainable machine learning for phishing site detection: A high-efficiency approach using boosting models and SHAP', The Journal of Engineering, 2025(1).

Wilcoxon, F. (1945) 'Individual comparisons by ranking methods', Biometrics Bulletin, 1(6), pp. 80–83.

Zhang, X., Zhao, J. and LeCun, Y. (2015) 'Character-level convolutional networks for text classification', in Advances in Neural Information Processing Systems (NeurIPS), pp. 649–657.

Zhou, J., Zhang, K., Zheng, B., Zhou, Y., Xie, X., Jin, M. and Liu, X. (2025) 'A malicious URL detection framework based on custom hybrid spatial sequence attention and logic constraint neural network', Symmetry, 17(7), art. 987.