

Beyond URL Analysis: Evaluating Content-Based Phishing Detection Using HTML Structural Features

Yousif A. Alhaj^{1,2,3}, Marwan M. Al-Falah^{*4}, E. S. Gebel⁴, Naif M. Al-Falah⁴, Abdulrahman Mohammed Hussein Obaid⁵, Mijahed Aljober¹, Hamzah A. M. Ali⁶, Ismail I. Al-Ahmed⁷, and Somaia Yaseen Ahmed Al-Gwaini⁸

¹Department of Artificial Intelligence, College of Engineering & Smart Computing, Modern Specialized University, Sana'a, Yemen, mijahed@msu.edu.ye

²Department of Artificial Intelligence, College of Engineering & Smart Computing, 21 September University of Medical and Applied Sciences, Sana'a, Yemen

³Department of Cybersecurity, Al-Saeeda University, Sana'a, Yemen

⁴Institute of Computer Science and Cybersecurity, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia, marwanmalfalah@gmail.com, gebel_es@spbstu.ru, alfalah.n@edu.spbstu.ru

⁵Department of Information Technology, 21 September University of Medical and Applied Sciences, Sana'a, Yemen

⁶College of Software and Technology, Zhejiang University, China

⁷Department of Information Technology, Al-Saeeda University, Sana'a, Yemen

⁸Department of Information Technology, Sana'a University, Sana'a, Yemen

Abstract For more than two decades, phishing pages have continued to evade defensive systems through increasingly sophisticated techniques. Most existing approaches combine URL features, telemetry, reputation signals, and HTML content, making it difficult to isolate the contribution of HTML alone. This study investigates whether phishing and legitimate pages can be distinguished using only HTML-derived content features. A balanced dataset of 40,000 pages was constructed through live HTTP retrieval, including 20,000 phishing pages from the verified PhishTank feed and 20,000 legitimate pages from the Tranco top-domain list. Using BeautifulSoup4, each page was represented by 15 HTML-derived features without using lexical URL features, reputation information, or network telemetry. Seven machine-learning classifiers were evaluated using 5-fold stratified cross-validation. Random Forest achieved the best performance with a positive-class F1 score of 0.940 ± 0.002 and a ROC-AUC of 0.983 ± 0.001 , while producing a 6.5% false-negative rate and a 5.3% false-positive rate. Feature-importance analysis showed that `n_links`, `n_scripts`, `n_images`, and `title_len` accounted for approximately two-thirds of the overall importance ranking. An ablation study demonstrated that the five most important features achieved an F1 score of 0.926, 1.5 percentage points below the All-15 ablation result. SHAP analysis and error inspection revealed that structurally rich phishing pages can closely imitate legitimate websites. The findings demonstrate that static HTML structure provides substantial discriminative information for phishing detection while revealing important limitations of content-based approaches. No claims are made regarding temporal, cross-dataset, or adversarial generalization.

Keywords: Phishing Detection, Content-Based Detection, HTML Feature Extraction, Machine Learning, Model Interpretability, Cybersecurity

*Corresponding author: marwanmalfalah@gmail.com. ORCID: 0000-0003-4142-6932

1 Introduction

Phishing dates to the mid-1990s and remains a widespread threat. The Anti-Phishing Working Group recorded 963,994 phishing attacks in the first quarter of 2024 and reported nearly five million attacks during 2023 [1]. Verizon’s 2024 Data Breach Investigations Report also identifies social engineering as a major breach pattern and discusses phishing as an important component of that pattern [2]. However, over time, phishing-detection techniques have diversified substantially: URL classifiers, hybrid systems using URL string and HTML syntax, deep learning on entire raw documents, screenshot and visual similarity engines, with an impressive 95% or more accuracy reported on the published benchmark split [3].

Large language models are also reshaping the broader phishing ecosystem by enabling scalable and adaptable textual phishing campaigns [4]. Although such campaigns are outside the webpage-classification scope of this study, they reinforce the need to evaluate detection methods against rapidly evolving attack-generation techniques.

That is, the published numbers have a subtle problem. Where recommended systems combine classifying URL lexical patterns, structural prediction of HTML syntax, and reputation clues into a single model, where the model performs well, it is difficult to separate the effectiveness of each signal. URL patterns tend to dominate since phished site URLs also bear typical lexical fingerprints, such as paypal.com instead of paypal.com, shortened bit.ly links to the victim's site, or punycode domains exploiting the confusion of Unicode characters. This makes the apparent contribution of HTML content alone difficult to isolate from the published numbers. This is not a mere theoretical concern. Consider a corporate security gateway that only gets to see the page rendered at the end of the redirect chain but never the page's originating URL. Or a content-filtering service analyzing pages given to it whose URLs have already been rewritten to click-tracking service URLs. In either case, the URL signal is partially or wholly unavailable, forcing the system to rely on the rendered content.

This paper addresses a focused empirical question: what can and cannot be inferred for phishing-page classification from HTML-derived content features alone? We do not propose a new architecture. The goal is not to establish a new benchmark record on a benchmark. What we propose is a clear and interpretable assessment of what static HTML structure can reveal when it is the only signal that a classifier is allowed to see, applied on a live-retrieved balanced-class dataset of 40,000 pages. The contribution can be summarized as follows:

- 1) We performed live HTTP retrieval of a balanced 40,000-page dataset (20,000 phishing pages from PhishTank's verified feed online and 20,000 legitimate pages from the Tranco top-100,000 domain list) and published the full methodology, including timeout values, checkpoint-resume behavior, deduplication strategy, final class counts, and logged failure modes.
- 2) We evaluated seven supervised classifiers under supervised learning-Gaussian NB, SVM-RBF, KNN, Decision Tree, Random Forest, AdaBoost, and MLP using a 15-dimensional feature vector derived solely from HTML text-source, sampled using constant-speed retrieval, while explicitly rejecting URL lexical features, reputation lookup, or network telemetry.
- 3) Rather than relying only on mean accuracy, we report positive-class precision, recall,

F1, fold-level standard deviations, ROC-AUC, false-negative rates, false-positive rates, and aggregated out-of-fold confusion matrices.

- 4) We perform feature-level interpretability analyses, from permutation importance, global SHAP, class-specific SHAP (revealing directional asymmetries not seen in the global view), dependence plots, ablation using smaller subsets of features, and qualitative error analysis on the actual misclassified pages.

The remainder of the paper is organized as follows: Section 2 reviews related work and positions our contribution in this context. Section 3 describes the methodology, Section 4 presents the experimental results, Section 5 discusses their implications, Section 6 outlines the limitations and future work, and Section 7 concludes the paper.

2 Related Work

The survey of literature on phishing detection leads one to believe that methods currently being used might be referred to as the several generations of phishing detection. After deploying blacklists and heuristic rules in the early 2000s, the field turned to classical machine learning in the early 2010s and has lately witnessed the proliferation of deep learning across many input modalities. Of the several streams of literature, four have the closest connection to the present work.

Recent systematic and bibliometric reviews confirm this progression and identify feature engineering, hybrid architectures, deep learning, and explainability as recurring themes in contemporary phishing detection research [5, 6].

2.1 URL-Based And Hybrid Methods

URL-based learning classifiers mark the first wave of ML-based phishing detection. Lexical features, such as URL length, average characters per token, character entropy, the density of special characters, brand-imitating substrings, suspicious top-level domains, etc., are derived from the URL then fed into a classifier. Feature-engineered URL benchmarks remain common in this area. For example, Linh and Hung introduced a dataset of 111,660 benign and phishing URLs enriched with 22 numerical lexical and structural features for machine-learning and language-model evaluation [7]. For resource-constrained mobile environments, Rao et al. combined handcrafted URL features, transformer embeddings, and multiple learners in a super-learner ensemble [8]. The models show the best performance within the distribution of training data. For instance, a cross-dataset evaluation highlights the unstable nature of URL-based classifiers when tests involve cross-source data. According to Mia et al. [9], such classifiers can lose nearly 20 points on the F1 score when applied to different test collections. Thus, research has led the authors to advocate using hybrid classifiers that use features based on URLs combined with HTML structure and resource-based information [10, 11]. Although these are more robust classifiers, they are more challenging to interpret due to the convolved effect of different sources of information on the final classification. Network-level alternatives have also emerged. GraPhish represents encrypted TLS traffic as graphs and demonstrates that traffic metadata can provide a strong phishing signal without inspecting the webpage content itself [12]. Such network telemetry is deliberately excluded from our HTML-only evaluation.

2.2 Content-Based Detection

Many publications demonstrated that the HTML structure could encode relevant phishing signals. Credential input fields, banking application imitation, and multiple redirects represent typical structural features that go beyond the capabilities of pure URL-based features. Privacy-preserving techniques based on hashed representations of HTML documents have made this point. However, most of these works choose a deep-learning-based approach to entire documents rather than handcrafted features [10]. Multi-modal HTML analysis was also used by Çolhak et al. [13]. The majority of these studies focus on overall accuracy, and few provide details of the feature contribution in classification or details per class. More recently, Goldenits et al. evaluated 15 small language models for phishing detection using raw HTML and URLs and highlighted trade-offs among classification performance, inference cost, and local data privacy [14]. This direction is complementary to the present study: language models consume rich raw-document representations, whereas our analysis isolates the predictive value of a compact and interpretable 15-feature vector.

2.3 Deep Learning Baselines

Deep models typically outperform classic baselines that have been used for phishing detection so far. For example, via a deep feed-forward model, Zara et al. [3] achieved above 95% accuracy with features derived from HTML and URLs. Similarly, with richer features, high-resolution images, and state-of-the-art architectures in both EfficientNet and BERT domains, Murhej and Nallasivan [11] achieved an accuracy higher than 96%. It is not surprising that our best F-measure of 0.940 with 15 manually selected HTML features does not reach these performances. Deep models learn features from raw document representations. Our features are condensed representations: 15-dimensional vectors. In addition, most deep baselines use URL-lexical features, which we purposely excluded. Our goal was to determine the limits of classical-ML style phishing detection using only HTML content.

Visual-similarity systems form another important family of deep phishing detectors. Wang et al. combine global webpage appearance with local logo information, whereas Ji et al. show through a large-scale real-world evaluation that visual detectors can degrade when reference coverage changes or visual elements are manipulated [15, 16]. Recent hybrid frameworks have also combined GAN-based augmentation, CNN-GRU-Transformer feature learning, optimization-based feature selection, and SHAP explanations [17].

2.4 The Interpretability Gap

Interpretability remains less consistently developed than predictive performance reporting. Mia et al. [9] considered post-hoc explanations for URL features, and recent hybrid deep-learning work has incorporated SHAP-based explanations [17]. However, per-class explanations, feature ablation, and qualitative inspection of actual misclassified webpages are still rarely reported together. We address this interpretability gap by running an array of interpretability tests at the proposed dataset and classifier levels by using permutation importance, global SHAP, class SHAP, dependence plots, reduced feature subsets by ablation, and qualitative error analysis. Table 1 positions our work against some of the prior works that guided us.

Table 1: Comparison with Representative Prior Work on Phishing Detection

Study	Dataset	Features	URL?	HTML?	DL?	Interpret.	Eval.	Result
Zara et al. [3]	Custom crawl	HTML + URL	Yes	Yes	Yes	No	Train/Test	>95% Acc
Çolhak et al. [13]	PhishTank+Alexa	HTML content	No	Yes	Yes	No	CV	~97% Acc
Murhej et al. [11]	Custom	Combined	Yes	Yes	Yes	No	Train/Test	>96% Acc
Mia et al. [9]	Multiple	URL	Yes	No	No	XAI	Cross-data	~20% drop
This work	PhishTank+Tranco	HTML only (d=15)	No	Yes	No	SHAP+Abl.	Strat. 5-fold	F1=0.940

3 Methodology

3.1 Problem Definition

The problem the modern classifier tackles is a two-class classification problem: to decide if a given observed Web page is phishing or legitimate. From raw HTML, a page is represented by a 15-dimensional feature vector, $x \in \mathbb{R}^{15}$. The classifier learns the mapping $f(x) \rightarrow y$, where $y = 1$ means phishing and $y = 0$ means legitimate. The feature dimensionality is kept fixed at $d = 15$ in the entire paper. Importantly, the pipeline does not use lexical features of the page address, reputation lookups, or network-derived telemetry. URLs appearing inside HTML hyperlinks are used only to compute aggregate hyperlink features such as external-link counts and external-domain ratios. The training and testing samples come from the same collection window, so our reported results characterize within-distribution performance. We do not test temporal out-of-distribution generalization, cross-dataset transfer, or adversarial robustness. We explicitly highlight these as future research directions in Section 6.

3.2 Data Collection

Phishing samples come from PhishTank's verified online feed, "online-valid.csv," while legit samples originate from the Tranco list of top domains. Both lists were collected at almost the same time, during a single window. The final notebook run records the collection timestamp as 16 April 2026. Actual page content collection was implemented using Python's requests library (v2.31) for HTTP and BeautifulSoup4 (v4.12) with lxml as the parsing backend for the HTML DOM. Configuring the fetcher, we set the connection timeout to 5 seconds, the read timeout to 10 seconds, retried failed requests only once, and inserted a very small randomized sleep between each request to avoid putting too much load on any single site. A simple fetcher could look like this as in Listing 1.

Listing 1: Reproducing the analytical workflow

```

1 def safe_fetch_html(url):
2     try:
3         r = requests.get(url, timeout=(5, 10),
4                           headers=HEADERS,
5                           allow_redirects=True)
6         if r.status_code != 200 or len(r.content) < 256:
7             return None, None, "bad_response"
8         return r.text, r.url, None
9     except RequestException as e:
10        return None, None, type(e).__name__

```

Thus, this notebook outputs 40,000 pages; 20,000 of those pages are Phishing and 20,000 are Legitimate. In contrast, for this notebook, the 56,492 PhishTank candidate URLs and 200,000 candidate URLs derived from Tranco were used. Because collection is

checkpointed and resumed, when resumed, the run will pick up from where the checkpoint left off on the legitimate side and continue with the remainder of Phishing collection. Therefore, the final notebook's logging metadata reports 9,169 logged failures on the Phishing side and 0 on the Legitimate side for the resumed run, while the number of attempted/failed Legitimate source retrievals is never recalculated on the resumed run. For this reason, Table 2 reports the final class counts, candidate-pool sizes, and logged failures as reported by the final notebook's logging metadata but does not report the retrieval rates for the Legitimate sources.

Table 2: Final Dataset Construction Summary from the Checkpoint-Resumed Notebook Run

Stage	Phishing	Legitimate
Candidate URLs available	56,492	200,000
Logged failures in resumed notebook run	9,169	Not re-estimated
Successful retrievals	20,000	20,000
Final samples used	20,000	20,000

Balance means that the final data set is exactly 50:50 so that the common class imbalance confounds normally present in this field of research which would force us to interpret the results of previous literature cautiously is avoided. This exact 50:50 data distribution was replicated in each fold via stratified 5-fold cross-validation. No SMOTE, class weighting, or any other balancing rebalancing technique was applied. All continuous features are z-score normalized by parameters learned from the training samples of a fold obtained via stratified 5-fold cross-validation. Features are not scaled on the whole data set before splitting to avoid leakage.

3.3 Feature Extraction

In total, 15 features are extracted for each page: four related to textual content, eight corresponding to structural properties, and three linked to hyperlinks. Table 3 details all features and their derivation.

Table 3: The 15 HTML-Derived Content Features ($d = 15$)

#	Feature	Category	Type	Definition / Extraction Rule
1	n_links	Hyperlink	Int	Count of all <a href> elements
2	n_scripts	Structural	Int	Count of <script> tags (inline + external)
3	n_images	Structural	Int	Count of tags
4	title_len	Textual	Int	Character length of <title> tag content
5	n_iframes	Structural	Int	Count of <iframe> tags
6	text_len	Textual	Int	Character count of visible body text
7	word_count	Textual	Int	Whitespace-separated tokens in visible text
8	n_inputs	Structural	Int	Count of <input> elements (any type)
9	n_external_links	Hyperlink	Int	Links pointing to a different registered domain
10	keyword_hits	Textual	Int	Count of distinct phishing keywords present
11	external_link_ratio	Hyperlink	Float	$n_external_links / n_links$
12	n_forms	Structural	Int	Count of <form> elements
13	has_email_field	Structural	Bin	1 if <input type='email'> exists
14	has_submit_field	Structural	Bin	1 if submit button exists
15	has_password_field	Structural	Bin	1 if <input type='password'> exists

A few extraction details are worth spelling out. Visible text is pulled out only after script, style, noscript, and elements with display:none are stripped from the soup, otherwise the text_len and word_count counts get inflated by JavaScript code and CSS rules

that the user never actually sees. External links are identified by comparing the registered domain of each href against the page's own registered domain via tldextract, not by raw netloc string matching which would fail on subdomain variants like login.example.com and example.com being incorrectly treated as different sites. The phishing keyword list contains eight terms: verify, login, password, update, bank, account, confirm, secure. We count distinct presence per page (whether each keyword appears at least once) rather than total occurrences, so a page that contains the word "verify" thirty times still scores 1 for that keyword, not 30. Binary features stay as 0/1 indicators with no normalization z-scoring a binary variable produces a meaningless transformation.

The core extraction loop is, simplistically, as shown in Listing 2.

Listing 2: Simplified HTML feature extraction workflow

```

1 def extract_features(seed_url, html, final_url=None):
2     soup_dom = BeautifulSoup(html, "lxml")
3     # Structural features
4     n_scripts = len(soup_dom.find_all("script"))
5     n_images = len(soup_dom.find_all("img"))
6     n_forms = len(soup_dom.find_all("form"))
7     n_iframes = len(soup_dom.find_all("iframe"))
8     # Hyperlink features
9     hrefs = [
10         a.get("href", "")
11         for a in soup_dom.find_all("a")
12         if a.get("href")
13     ]
14     n_links = len(hrefs)
15     # Textual features
16     visible_text = extract_visible_text(html)
17     text_len = len(visible_text)
18     word_count = len(visible_text.split())
19     return feature_vector

```

This pseudocode intentionally separates the DOM used for structural counting from the DOM used for visible-text extraction. Structural tags are counted before any text-cleaning deletion is applied; deletion is applied only to the text-specific copy of the DOM.

3.4 Models and Evaluation

Following the most commonly applied taxonomy, the study covers seven supervised classifiers: the Gaussian Naive Bayes, the RBF kernel Support Vector Machine, the k-Nearest Neighbor, the Decision Tree, the Random Forest, the AdaBoost, and a Multi-Layer Perceptron. All of them are based on the defaults of scikit-learn 1.8, except for two explicitly disclosed modifications to the default parameter setting: the instantiation of the Support Vector Classification (SVC) allows for probability=True to compute ROC-AUC on the outputs, and the MLPClassifier forces 500 max_iter instead of the default 200, which is sometimes too short for convergence on this data set before hitting the iteration threshold. Both Random Forest and AdaBoost use n_estimators=100. We do not use grid or random search for hyperparameter tuning. This deserves an explanation: on a fixed data set, hyperparameter tuning leads to an overly optimistic estimate of predictive performance, and more importantly complicates the interpretability of feature contribution, which we are particularly interested in here.

The models and the 5-fold cross-validation loop are defined As shown in Listing 3

Listing 3: Python implementation of the evaluation pipeline

```

1 models = {
2     "Gaussian_NB": GaussianNB(),
3     "SVM-RBF": SVC(
4         kernel="rbf",
5         probability=True,
6         random_state=42
7     ),
8     "KNN": KNeighborsClassifier(),
9     "Decision_Tree": DecisionTreeClassifier(
10         random_state=42
11     ),
12     "Random_Forest": RandomForestClassifier(
13         n_estimators=100,
14         random_state=42
15     ),
16     "AdaBoost": AdaBoostClassifier(
17         n_estimators=100,
18         random_state=42
19     ),
20     "MLP": MLPClassifier(
21         max_iter=500,
22         random_state=42
23     ),
24 }
25
26 skf = StratifiedKFold(
27     n_splits=5,
28     shuffle=True,
29     random_state=42
30 )
31
32 for name, clf in models.items():
33     pipe = Pipeline([
34         ("preprocess", preprocess),
35         ("model", clf)
36     ])
37     # Train-fold-only normalization

```

Performance is reported on average $\pm$ standard deviation over the 5 stratified folds. Additionally, precision, recall, and F1, unless otherwise noted, are computed with average='binary' and pos_label=1, following the clarification that the numbers reported were not macro-averaged but average metrics for the positive class. Along with these metrics, accuracy, ROC-AUC, false negative rate, and false positive rate are also reported. Confusion matrices are computed by summing over all out-of-fold predictions. About significance testing: for five folds, a two-sided exact Wilcoxon signed-rank test cannot yield $p < 0.05$. The smallest p-value attainable from this test, given this sample size, is 0.0625, which is obtained if the differences in all five folds come with the same sign. Instead of reporting p-values that look significant but are not, dominance counts per fold are reported for pairwise model comparisons; the inferential ceiling within which these counts are interpreted is acknowledged, as well.

3.5 Implementation And Reproducibility

The software environment was Python 3.11 with scikit-learn 1.8, pandas 3.0, numpy 2.4, BeautifulSoup 4.12, Requests 2.31, tldextract 5.1, and shap 0.51. Computation was performed on a MacBook Air M4 with a 16 GB unified memory configuration. For this particular study, data collection, seven-parallel model runs, and later interpretability analyses amount to an elapsed wall-clock time of approximately 18 h, where the actual time taken for live retrieval dominates. All code, requirements.txt, README, and the full results (CSV tables; PNG and PDF figures; metadata.json) are shared on the GitHub repository [18] and have been archived at Zenodo under DOI 10.5281/zenodo.20315649 [19]. The random seed is fixed at 42 throughout. Train-fold-only normalization is done with scikit-learn's ColumnTransformer inside the CV loop, thus keeping the definition of the test fold pristine.

The analytical workflow can be reproduced as as in Listing 4.

Listing 4: Reproducing the analytical workflow

```

1 git clone \
2 https://github.com/MarwanMAlfalah/content-phishing-detection.git
3 cd content-phishing-detection
4 python -m venv venv
5 source venv/bin/activate
6 pip install -r requirements.txt
7 jupyter notebook \
8 Content-Based-Phishing-Detection_v6_local.ipynb

```

4 Experimental Evaluation

4.1 Overall Performance

Table 4 reports cross-validation performance for all seven classifiers. Random Forest takes the top spot on F1 (0.940 ± 0.002) and ROC-AUC (0.983 ± 0.001) with the lowest variance among the strong performers. Decision Tree comes second on F1 at 0.921, then KNN at 0.913 and MLP at 0.908. The fold-level standard deviations are small, ranging from 0.001 to 0.004 across the reported metrics. We report this variability descriptively and restrict the interpretation to within-distribution 5-fold cross-validation. Gaussian Naive Bayes anchors the bottom: it has the highest recall in the table (0.973), but a 0.349 false-positive rate, which makes it operationally unusable in any deployment scenario. Flagging one in three legitimate pages as phishing would generate so many false alarms that the classifier would be impractical in operational settings.

Table 4: Performance Comparison (Mean $\pm$ SD, Positive-Class Metrics, 5 Folds)

Model	Acc	Prec	Recall	F1	AUC	FNR	FPR	F1 SD	AUC SD
Random Forest	0.941	0.946	0.935	0.940	0.983	0.065	0.053	0.002	0.001
Decision Tree	0.920	0.912	0.930	0.921	0.929	0.070	0.089	0.002	0.002
KNN	0.914	0.927	0.899	0.913	0.956	0.101	0.071	0.004	0.002
MLP	0.904	0.875	0.943	0.908	0.970	0.057	0.135	0.002	0.001
SVM-RBF	0.889	0.852	0.942	0.894	0.945	0.058	0.164	0.004	0.003
AdaBoost	0.885	0.841	0.949	0.892	0.957	0.051	0.179	0.004	0.003
Gaussian NB	0.812	0.736	0.973	0.838	0.885	0.027	0.349	0.003	0.004

4.2 Descriptive Comparison Among Models

Random Forest beat every other classifier on F1 in all five folds—a perfect 5/5 against each competitor. With only five paired observations per comparison, the exact two-sided Wilcoxon test produces $p = 0.0625$ in every case, the smallest p-value the test can produce at this sample size. The Random Forest advantage is consistent and substantial in magnitude (multi-percentage-point gaps in mean F1), but we stop short of claiming formal significance at $\alpha = 0.05$. A 10×5 repeated stratified k-fold protocol, which would generate the 50 paired observations a proper significance test needs, is something we recommend for follow-up work.

4.3 ROC Analysis

Figure 1 plots ROC curves for all seven classifiers. The curves are computed from concatenated out-of-fold predictions across the five folds rather than the arithmetic mean of per-fold curves the difference matters because per-fold averaging can hide rare-class behavior at the curve extremes. Random Forest dominates over the entire range of operating thresholds. The MLP curve approaches Random Forest at high false-positive rates but lags at the operating points that matter most for deployment, where a phishing detector typically lives at a low FPR.

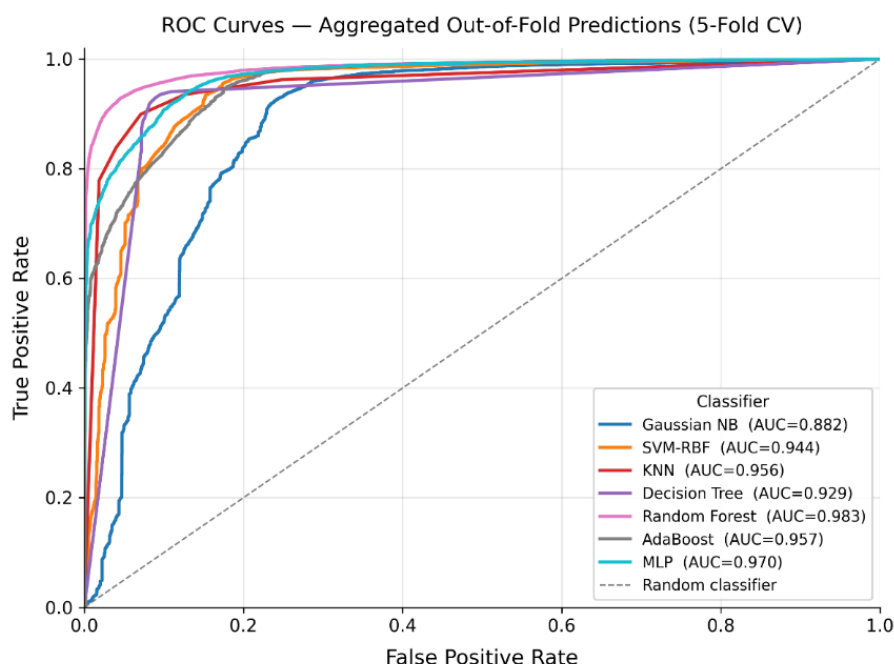


Figure 1: ROC curves for the seven classifiers, computed from aggregated out-of-fold predictions across the five stratified cross-validation folds. AUC values are shown in the legend.

4.4 Aggregated Confusion Matrices

Table 5 shows aggregated confusion-matrix counts from all 40,000 out-of-fold predictions, normalized to $N = 600$ per model so the rows are directly comparable. The metrics in the right-hand columns are computed from the full raw counts. Random Forest produces the

most balanced error distribution: 16 false positives and 20 false negatives per 600 samples. Gaussian NB illustrates the cost of recall-only optimization only 8 false negatives, but 105 false positives.

Table 5: Aggregated Confusion Matrix (Counts Normalized to N=600; Metrics from Full Dataset)

Model	N	TN	FP	FN	TP	Acc	Prec	Recall	F1	FPR
Rnd. Forest	600	284	16	20	280	0.941	0.946	0.935	0.940	0.053
Dec. Tree	600	273	27	21	279	0.920	0.912	0.930	0.921	0.089
KNN	600	279	21	30	270	0.914	0.927	0.899	0.913	0.071
MLP	600	260	40	17	283	0.904	0.875	0.943	0.908	0.135
SVM-RBF	600	251	49	17	283	0.889	0.852	0.942	0.894	0.164
AdaBoost	600	246	54	15	285	0.885	0.841	0.949	0.892	0.179
Gaussian NB	600	195	105	8	292	0.812	0.736	0.973	0.838	0.349

4.5 Feature Importance

Permutation importance for Random Forest is reported in Table 6.

Table 6: Permutation Feature Importance (Random Forest)

Rank	Feature	Importance (Mean)	Importance (SD)
1	n_links	0.200	0.002
2	n_scripts	0.141	0.002
3	n_images	0.082	0.001
4	title_len	0.081	0.001
5	n_iframes	0.057	0.001
6	text_len	0.042	0.001
7	word_count	0.029	0.001
8	n_inputs	0.027	0.001
9	n_external_links	0.023	0.001
10	keyword_hits	0.014	0.000
11	external_link_ratio	0.013	0.000
12	n_forms	0.008	0.001
13	has_email_field	0.002	0.000
14	has_submit_field	0.002	0.000
15	has_password_field	0.001	0.000

4.6 Class-Specific SHAP Analysis

We computed SHAP values for the Random Forest model using 2,000 randomly sampled transformed feature vectors. Figures 2 and 3 show the SHAP summaries for the phishing-class output (class = 1) and the legitimate-class output (class = 0), respectively. Both plots are based on the same 2,000-sample subset and therefore explain the two output classes of the binary classifier rather than two disjoint subsets of phishing and legitimate webpages.

Because the class probabilities in a binary classifier are complementary, the two SHAP summaries exhibit approximately mirrored directional patterns. They are presented together to make explicit how the same feature values shift the model output toward phishing or toward legitimacy.

In the case of phishing-output mean absolute SHAP, the top three features are `n.links`, `word_count`, and `text_len`, followed by `n.images`, `n.scripts`, `n.external.links`, `title_len`, and `external_link_ratio`. This differs from permutation importance, where the top four features are `n.links`, `n.scripts`, `n.images`, and `title_len`, respectively. This difference is expected since permutation importance measures how much feature permutation can cause performance loss, while SHAP measures the average magnitude of a feature's contribution to individual predictions.

Therefore, Figure 4 illustrates the dependence of the three highest-ranked features by phishing-output mean absolute SHAP value: `n.links`, `word_count`, and `text_len`.

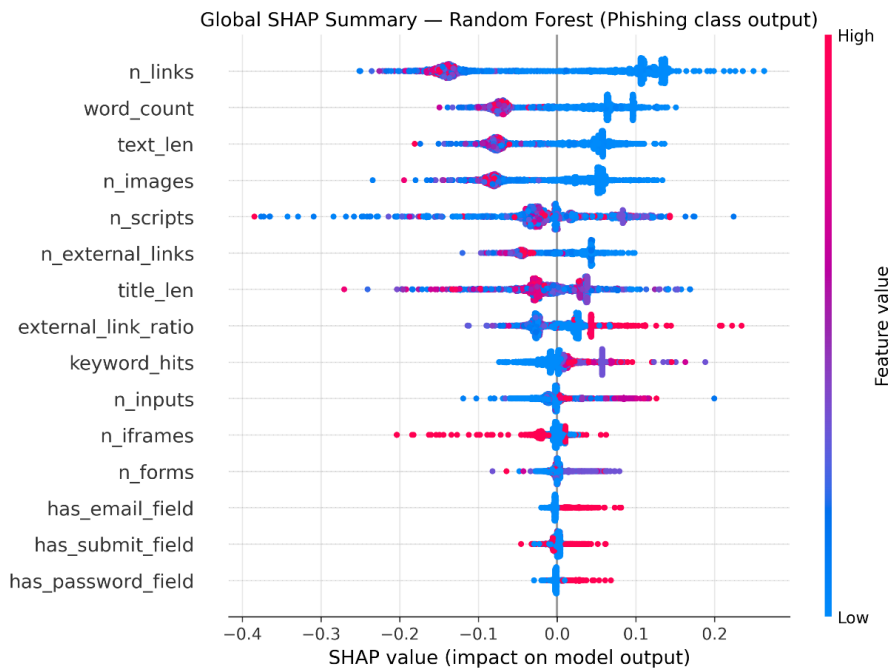


Figure 2: SHAP summary for the Random Forest phishing-class output (class = 1), computed on the 2,000-sample SHAP subset.

4.7 Ablation Study

If four features account for most of the importance, what about keeping only four (or three, or five) and discarding the rest? Table 7 shows Random Forest performance on subsets with a reduced number of features. Selecting only the first three features (`n.links`, `n.scripts`, `n.images`) achieves an $F1 = 0.908$, only 3.2 percentage points less than the model with 15 features. A Top-5 reaches an $F1 = 0.926$, approximately 1.5 percentage points below the All-15 ablation-loop result. The Top-8 subset reduces the gap to 0.4 percentage points. The marginal gain from the features beyond rank 8 is below 0.5 points of $F1$, which reflects what Table 6 already suggested about the concentration. In deployment scenarios where the speed of feature extraction is critical, parsing each `<input>` field for millions of pages per hour, the practical implication is that a 5-feature extractor achieves

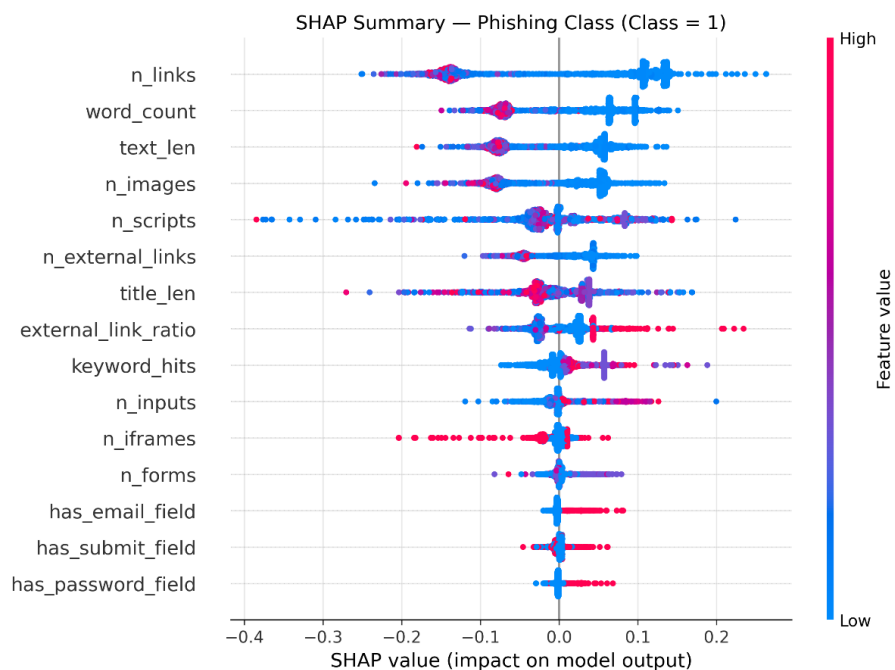


Figure 3: SHAP summary for the Random Forest legitimate-class output (class = 0), computed on the same 2,000-sample SHAP subset.

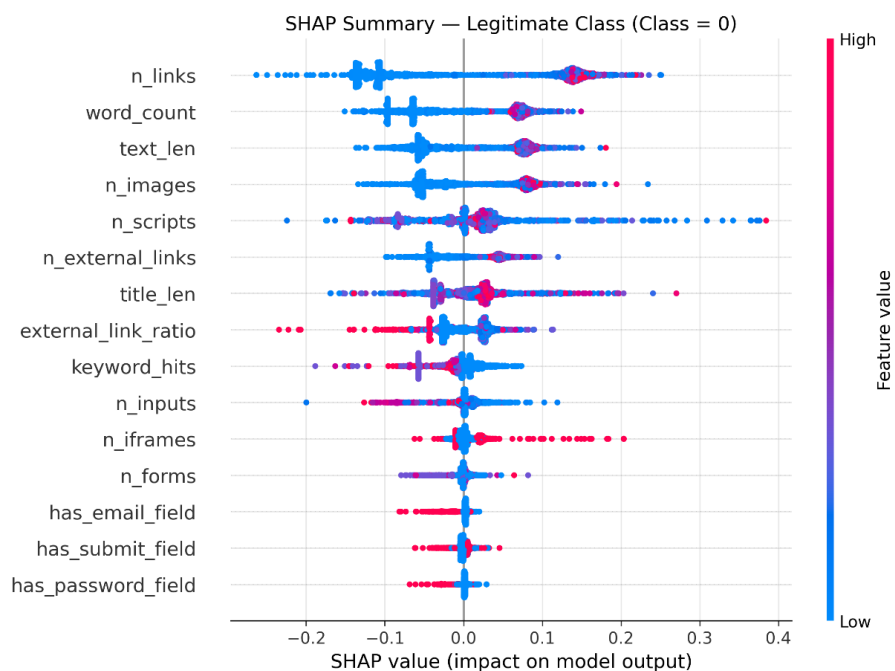


Figure 4: SHAP dependence plots for the top three features by phishing-output mean absolute SHAP value: n.links, word.count, and text.len

almost full performance at a fraction of the cost. Each panel in Figure 4 plots the z-normalized feature value on the x-axis and its SHAP contribution to the phishing-class output on the y-axis.

Table 7: Ablation Study: Random Forest with Reduced Feature Subsets

Subset	# Feat	Mean F1	F1 SD	Mean AUC	AUC SD
Top-3	3	0.908	0.002	0.970	0.001
Top-5	5	0.926	0.003	0.977	0.002
Top-8	8	0.937	0.002	0.982	0.001
All-15	15	0.941	0.002	0.983	0.001

4.8 Qualitative Error Analysis

Random Forest produced 1,306 false negatives and 1,061 false positives across all out-of-fold predictions. We pulled the actual misclassified pages and looked at them. The missed phishing pages (the false negatives) cluster around two structural patterns. The first pattern is sparse, login-style pages few links, almost no scripts whose minimalist footprint resembles plain legitimate landing pages. Picture a page that displays a copy of a banking login form with nothing else around it: no navigation menu, no footer, no sidebar. There is little structural information for the model to latch onto. The second pattern is the opposite extreme: phishing pages that mirror real sites with many images, scripts, and external CDN links, becoming statistically indistinguishable from genuine portals. These pages have clearly been built by attackers who copied the entire DOM of a target site rather than constructing a stripped-down clone.

The 1,061 false positives have their own pattern. Most are legitimate pages with unusually heavy resource footprints: news sites with dozens of third-party trackers, e-commerce pages pulling assets from many CDNs, and login portals whose structure happens to overlap with phishing-kit templates. Figure 6 visualizes mean feature values across the FN, FP, and TP groups, which confirms these patterns at a glance. False negatives have markedly higher text.len, word.count, and n.links than correctly detected phishing pages, supporting the observation that missed phishing pages are structurally richer than the typical phishing footprint our model expects.

Figure 4 — SHAP Dependence Plots (Top-3 Phishing Features)

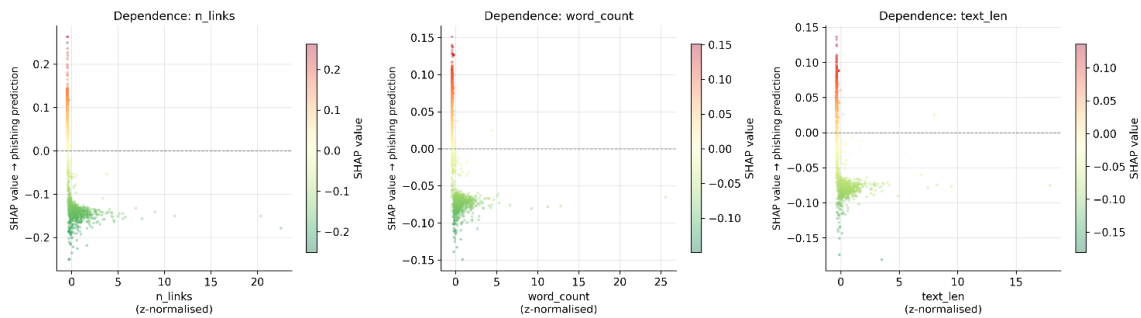


Figure 5: Mean feature values across three Random Forest prediction outcomes: false negatives (missed phishing), false positives (incorrectly flagged legitimate pages), and true positives (correctly detected phishing). Top features by permutation importance are shown

On computational cost: feature extraction with BeautifulSoup4 takes 50–150 ms per page on standard CPU hardware (we measured this on the M4 MacBook used for the experiments), and Random Forest inference on the 15-dimensional vector is under 1 ms.

The pipeline is light enough to serve as a pre-filter in production, sitting in front of more expensive deep-learning or visual-analysis stages.

5 Discussion

The first observation is that HTML-derived content features alone provided substantial predictive performance. Random Forest achieved an F1 score of 0.940 and an AUC of 0.983. Because the model excluded lexical URL features, reputation signals, and network telemetry, these results indicate that static HTML structure carries considerable discriminative information. However, the findings are limited to the balanced, within-distribution evaluation described in this study and should not be interpreted as deployment-ready performance estimates. The results instead support the use of HTML-based classification as a complementary layer alongside URL-, reputation-, network-, or visual-analysis methods.

The second point is that such a simple set of features carried almost all, if not all, of the predictive power for discrimination. The number of links, script files, and images loaded (`n_links`, `n_scripts`, `n_images`), supplemented only by one simple textual feature (`title_len`), gave us all the information we needed. Surprisingly, the intuitive credential-harvesting binary indicators became almost useless in this setup. Why does this happen? The way we interpret this is that, when dealing with real-world phishing pages, the difference between legitimate pages and phishing pages is more stable and robust with respect to their structural footprint: how many objects does the page have to load, and what is the shape of the layout more complex or more straightforward? In contrast, the presence of credential input fields may not vary so simply and more stably. For example, benign login portals do have password fields, submit buttons, and email inputs. In this case, the presence or absence of a binary feature like `has_password_field` asks a very narrow question "does this page have a password input?" The answer would be positive for both `bank.com` and `bank-secure-update.com`. On the other hand, the structural count features ask rich questions: "How many resources does this page have to load? How many places does it want the user to click?" A strong practical implication of this is that very simple 5-feature extractors can easily achieve an F1 score as high as 0.926, making them extremely attractive from the point of view of very light, resource-constrained deployment.

The third observation concerns the asymmetric class-output SHAP behavior. These very same features pull the output towards one class when the true class is phishing but towards the other class when the true class is legitimate. This explains, at least in part, the error types in the confusion matrix. Phishing pages mimicking the structural complexity of legitimate sites are misclassified as legitimate because they fall inside the feature space defined by typical legitimate website statistics. However, a few legitimate sites with unusually heavy resource loads are misclassified as phishing for the converse reason. This prior on structure fits most cases well, but it fails rather predictably for the two tail cases of the legitimate and phishing site distributions. This is one type of error hidden in a global SHAP summary but very visible once the summary is class-specific. Moreover, practitioners should note that a proxy evasion attack to the extent that a phishing site tries to replicate the complexity of a legitimate site would be misclassified by such a detector in this modality and would hence require complementary signals, e.g., visual similarity and network behavior.

What we bring to the field is not a new state-of-the-art model but rather a clear and well-interpretable image of HTML content in its pure form: how well it performs, which features do the lion's share of the work, what the error patterns are, and what the trade-

offs are. These empirical works often form the prerequisite for informing further studies of feature engineering and deployment steps but are normally absent from the reported architectural innovation.

6 Limitations and Future Work

We do not think these results generalize for a few reasons. The data is balanced 50:50 for reasons of design, thus taking away a confound of class imbalance while obscuring the true ratio of webpages surfacing on the open Web. The resulting performance metrics cannot be directly compared to practically deployed ones in which the prior probability of a page being phishing is much less than 50%. Live retrieval also imposes survivorship bias: short-lived phishing pages are less likely to remain accessible at collection time. Because the checkpoint-resumed run did not preserve complete attempt and failure totals for all retrieval stages, we do not report an overall retrieval-success rate. The resulting corpus may therefore overrepresent longer-lived phishing infrastructure.

These results have a clean baseline since hyperparameters were never tuned beyond the scikit-learn defaults to allow comparisons within this setting. Nested tuning is expected to change the ranking, improving SVM-RBF and MLP results more than other models. We neither conducted a temporal validation approach (older to newer) nor a cross-dataset or adversarial perturbation experiment. This limitation is important because recent adversarial evaluations have shown substantial detection-rate degradation in several task-specific phishing detectors when tested on deliberately modified webpages. Multimodal language-model-based detection was more robust in those experiments, but it remained susceptible to evasion [20]. Limitations of statistical inference apply, given the five-fold design structure, as discussed in Section 4.2. Most of the important work on features has concentrated on static HTML: features are generally tabulated from the HTML source of a page. Web pages that are driven primarily by JavaScript and whose important content is dynamically constructed via scripting may not be captured well in this representation. If a page builds its entire document object model (DOM) via React after the initial HTML fetch, it would appear close to empty with our extractor.

Future extensions of the present work include temporal validity as conducted through multiple years of PhishTank archives, cross-dataset transfer to other phishing repositories, adversarial test sets probing the structural prior for modern attack tooling, e.g., LLM-generated phishing pages, headless browser or scripted rendering pipeline evaluation, and Selenium and Playwright-based 10×5 repeated CV with nested hyperparameter tuning and deployment-realistic class imbalance handling.

7 Conclusion

This focused empirical characterization of HTML-derived features for phishing-webpage detection shows that, on a balanced dataset of 40,000 pages, Random Forest achieved an AUC of 0.983 and a positive-class F1 score of 0.940 using only 15 content features, without lexical URL features, reputation signals, or network telemetry. Permutation importance and SHAP analyses showed that a small set of structural and textual features contained most of the predictive signal, whereas the three credential-harvesting indicators contributed little additional information. In the ablation study, the Top-5 subset achieved an F1 score of 0.926, compared with 0.941 for the All-15 subset. Class-output SHAP analysis and qualitative error inspection revealed predictable failure modes: structurally

rich phishing pages may resemble legitimate websites, while unusually resource-heavy legitimate pages may occupy the same feature space as phishing pages. The contribution is therefore interpretive rather than architectural: the study isolates what static HTML structure can reveal, identifies the features carrying most of the signal, and documents where this modality fails. Claims of broader generalization require future temporal, cross-dataset, deployment-realistic, and adversarial evaluations.

References

- [1] Anti-Phishing Working Group. Phishing activity trends report: First quarter 2024. Technical report, Anti-Phishing Working Group, 2024.
- [2] Verizon. 2024 data breach investigations report: Executive summary. Technical report, Verizon Business, 2024.
- [3] U. Zara, K. Ayyub, H. Ullah Khan, A. Daud, T. Alsahfi, and S. Gulzar Ahmad. Phishing website detection using deep learning models. *IEEE Access*, 12:167072–167087, 2024.
- [4] F. Chen, T. Wu, V. Nguyen, and C. Rudolph. SoK: Large language model-generated textual phishing campaigns end-to-end analysis of generation, characteristics, and detection, 2025. arXiv preprint arXiv:2508.21457.
- [5] J. L. Wilk-Jakubowski, L. Pawlik, G. Wilk-Jakubowski, and A. Sikora. Machine learning and neural networks for phishing detection: A systematic review (2017–2024). *Electronics*, 14(18):3744, 2025.
- [6] D. Popescul and L. D. Radu. AI in phishing detection: a bibliometric review. *Frontiers in Artificial Intelligence*, 8:1496580, 2025.
- [7] D. M. Linh and T. C. Hung. A feature-engineered dataset of benign and phishing URLs for machine learning and large language models evaluation. *Data in Brief*, 63:112162, 2025.
- [8] R. S. Rao, C. Kondaiah, A. R. Pais, and B. Lee. A hybrid super learner ensemble for phishing detection on mobile devices. *Scientific Reports*, 15(1):16839, 2025.
- [9] M. Mia, D. Derakhshan, and M. M. A. Pritom. Can features for phishing URL detection be trusted across diverse datasets? a case study with explainable AI. In *Proceedings of the 2024 11th International Conference on Networking, Systems and Security, NSysS 2024*, pages 137–145, 2025.
- [10] J. Kim, Y. Kim, S. Lee, and S. Cho. Privacy-preserving phishing detection in HTML code using split learning. In *International Conference on ICT Convergence*, pages 173–178, 2024.
- [11] M. Murhej and G. Nallasivan. Component features based enhanced phishing website detection system using EfficientNet, FH-BERT, and SELU-CRNN methods. *Frontiers in Computer Science*, 7:1582206, 2025.

- [12] K. Manguli, C. Kondaiah, A. R. Pais, and R. S. Rao. GraPhish: A graph-based approach for phishing detection from encrypted TLS traffic. *Journal of Information Security and Applications*, 94:104216, 2025.
- [13] Furkan Çolhak, Mert İlhan Ecevit, Bilal Emir Uçar, Reiner Creutzburg, and Hasan Dağ. Phishing website detection through multi-model analysis of HTML content. In *Proceedings of International Conference on Theoretical and Applied Computing (ICTAC 2024)*, pages 171–184. Springer, 2025.
- [14] Georg Goldenits, Philip Koenig, Sebastian Raubitzek, and Andreas Ekelhart. Small language models for phishing website detection: Cost, performance, and privacy trade-offs. *Journal of Cybersecurity and Privacy*, 6(2):48, 2026.
- [15] M. Wang, L. Song, L. Li, Y. Zhu, and J. Li. Phishing webpage detection based on global and local visual similarity. *Expert Systems with Applications*, 252:124120, 2024.
- [16] Fujiao Ji, Kiho Lee, Hyungjoon Koo, Wenhao You, Euijin Choo, Hyoungshick Kim, and Doowon Kim. Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3201–3220, Seattle, WA, August 2025. USENIX Association.
- [17] Raheleh Ghadami and Javad Rahebi. An explainable hybrid deep learning-optimization framework for robust phishing attack detection using GAN and transformer-based feature learning. *Ain Shams Engineering Journal*, 16(12):103745, 2025.
- [18] M. Al-Falah. Beyond URL analysis: Evaluating content-based phishing detection using HTML structural features. GitHub Repository: <https://github.com/MarwanMAlfalah/content-phishing-detection>, 2026.
- [19] Yousif A. Alhaj, Marwan M. Al-Falah, E. S. Gebel, Naif M. Al-Falah, Abdulrahman Mohammed Hussein Obaid, Mijahed Aljober, Hamzah A. M. Ali, Ismail I. Al-Ahmed, and Somaia Yaseen Ahmed Al-Gwaini. Beyond URL analysis: Evaluating content-based phishing detection using HTML structural features, 2026. Zenodo Dataset, Version 1.0.0, DOI: 10.5281/zenodo.20315649.
- [20] Aditya Kulkarni, Vivek Balachandran, Dinil Mon Divakaran, and Tamal Das. From ML to LLM: Evaluating the robustness of phishing web page detection models against adversarial attacks. *Digital Threats: Research and Practice*, 6(2):1–25, 2025. Article 10.