International Journal of Advanced Multidisciplinary Application | IJAMA
Volume 3 Issue 11 Nov 2026
ISSN No: 3048-9350

Real-Time Phishing Website Detection Using Machine Learning: A Framework and Preliminary Benchmark

[1]Mitesh Mandawara and [2]Girdhari Lal
[1]Student, Department of Artificial Intelligence & Data Science, Poornima Institute of Engineering & Technology, Jaipur, Rajasthan, India
[2]Assistant Professor, Department of Artificial Intelligence & Data Science Poornima Institute of Engineering & Technology, Jaipur, Rajasthan, India
Email: mitessh21@gmail.com and girdhari.lal@poornima.org

Abstract*—Phishing websites imitate trusted web services to trick users into disclosing credentials, banking details and personal data. Blacklist and rule-based defences are effective against catalogued threats but respond slowly to newly created or modified sites, which motivates learning-based detection that generalises from URL and webpage characteristics. This paper presents a framework for real-time phishing website detection using machine learning, together with a preliminary benchmark of classical classifiers; the real-time framework and per-URL latency have not yet been evaluated. We formulate the task as binary classification over a URL and optional domain and webpage evidence, review the evolution of blacklist, heuristic, lexical, content-based, hybrid and adversarially robust detectors, and tabulate results reported by ten previous studies, clearly labelled as literature findings. We then propose the Real-Time ML-Based Phishing Detection Framework, a pipeline comprising URL validation and normalisation, tiered lexical, domain and webpage feature extraction, feature selection, classification with calibrated confidence, and feature-level explanation. An evaluation protocol is specified using accuracy, precision, recall, F1-score, ROC-AUC, confusion-matrix analysis and prediction time across candidate classifiers. The preliminary benchmark compares ten classical classifiers on the UCI Phishing Websites dataset (11,055 instances, 30 features) using a stratified 80:20 split and a held-out test set of 2,211 samples: Extra Trees achieved the highest accuracy (0.9774) and F1-score (0.9744), and Random Forest the highest ROC-AUC (0.9967), on data containing many duplicate rows. The complete framework, the deep-learning baselines and the proposed ensemble are not yet evaluated, and no performance is claimed for them.

Keyword: phishing detection, machine learning, URL analysis, feature engineering, ensemble learning, real-time detection, explainability, cybersecurity

1. INTRODUCTION

Phishing is a type of social engineering in which the attacker pretends to be someone trusted to convince victims to disclose sensitive data [1]. The attacker creates a fake website, mimics the look and function of a real service (like a bank, payment processor, webmail service or social network) and sends a link to it in an email, messaging app, social network or in an ad. If a victim clicks on the link, and enters a form, the information entered is sent directly to the attacker. Usernames and passwords, one-time pass codes, card and bank account numbers, government identifiers and other personal information are typical examples of the information targeted [2].

Phishing remains a serious issue in cyber-security, as it relies on human judgment not on software flaws, is low cost to scale using pre-made phishing kits, and is constantly tracked by industry groups like the Anti-Phishing Working Group [3]. Often, stolen credentials are the first phase of account takeover, financial fraud and larger-scale intrusions, meaning that one successful page can result in losses that exceed the effort put into creating it.

Blacklist and reputation services pre-identify URLs that have been previously determined to be malicious [4] and [5]. They have low false-positive rates, but are reactive: a page cannot be blocked until it is discovered, verified and published. Measurement studies reveal that listing delays and incomplete coverage can leave windows during which a user is exposed to a page [6]. An alternative approach to the list is to use a rule-based detector or a heuristic detector, which are based on hand-crafted conditions, such as an IP-address host, or an excessively large number of sub-domains, but fixed rules will need continuous upkeep and can be defeated once attackers learn them [2]. The most difficult case is when the phishing servers are newly created and somewhat customized, as campaigns are short-lived, domains are cheap to register, and attackers are able to use a compromised legitimate server or URL shortener to evade known indicators.

The advantage of machine learning (ML) is that it can identify statistical regularities from a set of labelled examples without being predefined, thus it can score URLs it has never seen before [7], [8]. There are measurable clues to look for that indicate a phishing URL or page: bizarre length of the URL; unusual character distribution; brand names in sub-domains; forms submitting to non-local hosts; or very new domain names. None of the traces is constituting a decisive signal, but supervised classifiers can be used to merge several weak signals and model non-linear interactions (especially tree ensembles), they can be retrained with a changing attacker behaviour.

The time of detection is also crucial. The vast majority of victims click on a link within minutes of receiving it, so in practice, a practical detector would need to provide a verdict before the page that was rendered, or before a form was submitted. This means that low latency is not an add-on: features that require DNS, WHOIS, search-engine or page-fetching calls do come with latency and published comparisons reveal significant differences between the various model families in terms of time and memory cost [9, 10].

Detectors are usually classified according to the evidence that they use. URL-based detectors work on the address string only, can be used before any network request, and can be used without compromising privacy, but can be fooled by well-formed URLs. Webpage-content based detectors check HTML, document object model (DOM), forms, scripts and resources, and offer more evidence but require a page to be downloaded. Hybrid detectors are a combination of views [11, 10].**Motivation and contribution.** Many published studies report high accuracy on a single static dataset while saying little about latency, explanation or robustness. This paper therefore specifies a detection framework designed around those requirements. Its contributions are: (i) a formal problem definition and measurable objectives; (ii) a structured literature synthesis with a comparison of ten previous studies; (iii) an original *Real-Time ML-Based Phishing Detection Framework* with tiered feature extraction, calibrated confidence and feature-level explanation; (iv) an inference algorithm and deployment design; and (v) an evaluation protocol with preliminary benchmark results for ten classical classifiers. **The complete framework, the deep-learning baselines and the proposed ensemble have not yet been evaluated, and no result is reported for them.**

2. PROBLEM STATEMENT AND OBJECTIVES

Let u denote a URL and e the optional domain or webpage evidence associated with it. A feature map φ produces a vector $x = \varphi(u, e)$, and a classifier outputs the probability $p(x)$ that the site is phishing. With decision threshold τ, phishing website detection is the binary classification task

$$\hat{y} = 1 \text{ (phishing) if } p(x) \geq \tau; \quad \hat{y} = 0 \text{ (legitimate) otherwise} \quad (1)$$

where 0 denotes a legitimate and 1 a phishing website. In a real-time setting the end-to-end time Textract + Tinfer must also remain below a latency budget Tmax.

The research objectives are to:

1) develop an ML-based phishing website detection framework;
2) extract useful lexical and structural features from URLs;
3) compare multiple ML algorithms under an identical protocol;
4) investigate URL-based and webpage/content-based feature combinations;
5) evaluate models using accuracy, precision, recall, F1-score and ROC-AUC;
6) analyse false positives and false negatives;
7) measure prediction latency for real-time deployment; and
8) identify a model that balances detection performance, robustness and computational cost.

Numerical performance is reported only for models that were actually trained and tested (Section 10); values for all other models remain to be measured.

3. LITERATURE REVIEW

This section summarises the evolution of phishing website detection. Numerical values quoted below are results *reported by the cited authors on their own datasets and protocols*; they are not directly comparable with one another and are not results of the proposed system.

3.1 Blacklist and Reputation-Based Detection

Blacklist systems look up a URL in a curated list of confirmed phishing sites, and whitelists do the opposite for trusted sites. PhishNet extended blacklists through predictive matching of URLs related to known entries [4], while comparative analyses of Google Safe Browsing, OpenPhish and PhishTank describe differences in coverage and update behaviour [5]. The PhishTime study measured the effectiveness of anti-phishing blacklists over time and highlighted the delay between a page appearing and being listed [6]. The advantages are speed and precision; the limitations are reactivity, incomplete coverage of new campaigns and dependence on continual list maintenance.

3.2 Rule-Based and Heuristic Detection

Heuristic detectors encode expert knowledge as rules over URL and page properties. CANTINA, for example, extracts high-weight terms from a page using TF-IDF and queries a search engine to judge whether the page matches its claimed identity [12]. Surveys catalogue many such URL, content and visual rules [2]. Rules are transparent and cheap to evaluate, but thresholds are brittle, require manual updating and are easy for adaptive attackers to circumvent.

3.3 URL Lexical Feature-Based Machine Learning

Lexical approaches derive features from the URL string alone. Common features include URL length, number of dots, hyphens and other special characters, number of sub-domains, suspicious keywords, use of an IP address as host, HTTPS-related indicators, character and word statistics, Shannon entropy and domain properties. Ma *et al.* combined lexical and host-based features to show that learned URL classifiers can flag malicious sites that blacklists miss [7]. Abutaha *et al.* generated 22 lexical features from a large labelled URL corpus, applied variance and multicollinearity filtering, and reported the best result with an SVM [13]. Sahingoz *et al.* used natural-language-processing (NLP) based word features and reported that random forest performed best [8]. Because many phishing sites now obtain valid TLS certificates, HTTPS presence is a weak indicator on its own.

3.4 Classical Machine Learning Approaches

Logistic regression (LR), naive Bayes (NB), decision trees (DT), random forest (RF) [14], support vector machines (SVM) [15], K-nearest neighbours (KNN), gradient boosting [16] and AdaBoost and XGBoost [17] are repeatedly benchmarked. Anakal et al. used seven such classifiers on a public Kaggle dataset and found RF to be the best and then deployed the best model via web interface [18]. Karim et al. assessed eight models and a voting hybrid on the same type of data [19] and Shahrivari et al. compared twelve classifiers [20] while Boyapati and Aygun reported XGBoost to be best on web-scraped features [10]. The general trend is that tree ensembles are good at tabular phishing features, however, the rankings vary depending on the dataset, the split and the feature set, and thus no model can be assumed as best before training.

3.5 Webpage and Content-Based Detection

Content-based detectors look at the structure of HTML, the DOM, form elements, hyperlinks, scripts, and iframes and external resources which are usually obtained through web scraping. Boyapati and Aygun pruned URL, network, domain, content and search-engine features and tested them using and without principal component analysis [10]. Phishpedia on the other hand identifies brand logos in page screenshots and cross checks with the claimed domain [21]. While more informative than just the URL, content analysis will incur latency, require downloading potentially malicious pages, and be susceptible to dynamic scripts and cloaking.

3.6 Ensemble and Hybrid Approaches

Ensembles combine classifiers or feature views whose errors are partly independent, which reduces variance and improves robustness. Karim *et al.* combined LR, SVC and DT by soft and hard voting with canopy feature selection [19]; Chiew *et al.* proposed a hybrid ensemble feature-selection framework in which feature subsets are ranked and combined before classification [22]; and Das Guptta *et al.* combined URL and hyperlink features, reporting that each set alone was insufficient [11]. The trade-off is greater training complexity and, depending on the design, higher inference cost.

3.7 Adversarial Phishing Detection

The inputs can be manipulated at test time to be misclassified by the detector, which is a form of attack [23] and the adversarial training is widely studied defence [24]. Multi-SpacePhish systematically investigated the evasion of ML detectors [25] in the feature, problem and mixed spaces. Duy et al. trained a Wasserstein GAN to generate adversarial domain names, reported evasion rates up to 88% for multiple models based on given URLs, and noted that multimodal models evaded less, and that retraining with adversarial samples led to detection rates exceeding 90% [9]. The results suggest the need for explicit robustness evaluation of URL-only detectors.

3.8 Real-Time Detection

There are a few studies that mention browser add-ons or web services. Abutaha et al. propose to include a URL-only classifier in the browser's middleware [13] and Anakal et al. present a trained model via a Streamlit application [18]. Although not very common, Duy et al. measured prediction time and memory for various detectors and found that multimodal detectors were significantly slower than unimodal detectors [9] and Boyapati and Aygun used a high-performance computing cluster for scraping [10]. Sharing this with real-world deployment, there's a need for tiered features, timeouts, caching, small models, and attention to the privacy of submitted URLs.

TABLE I
COMPARISON OF EXISTING PHISHING WEBSITE DETECTION RESEARCH

S. No.	Author(s)	Paper Title	Publication / Year	Dataset / Data Source	Features Used	Machine Learning Method	Reported Findings	Limitations
1	A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, S. R. K. Joga [19]	Phishing Detection System Through Hybrid Machine Learning Based on URL	IEEE Access, vol. 11, 2023	Kaggle phishing-URL dataset; 11,054 records with about 30 pre-extracted attributes (as reported)	URL-based attribute vectors (IP use, URL length, sub-domains, HTTPS, request/anchor URL, etc.)	DT, LR, RF, NB, GBM, SVC, KNN; hybrid LR+SVC+DT voting with canopy feature selection, grid search, cross-validation	RF reported as best single model (accuracy 96.77%); proposed approach reported at 98.12% accuracy; voting variants reported at 94.09–95.23%	Single static dataset with pre-extracted features; no latency study of feature extraction; results of the proposed model are not reported consistently across tables
2	P. T. Duy, V. Q. Minh, B. T. H. Dang, N. D. H. Son, N. H. Quyen, V.-H. Pham [9]	A Study on Adversarial Sample Resistance and Defense Mechanism for Multimodal Learning-Based Phishing Website Detection	IEEE Access, vol. 12, 2024	OpenPhish, PhishTank, Phishing Database, Alexa; domain set and 23,960-instance website set (as reported)	Domain strings; URL, HTML tags, content and JavaScript (multimodal)	15 ML/EL/DL/multimodal models; WGAN-based adversarial domain generator (AWG); adversarial retraining defence	Evasion up to 88%; SVM detection of adversarial samples as low as 10.02%; multimodal Shark-Eyes up to 99% detection; all detection rates above 90% after retraining	Short phishing-site lifespan limits data (per authors); multimodal models cost more time and memory
3	S. Anakal, K. Maka, A. Tadkal, S. Humanabad, S. Anakal, L. E [18]	Phishing Website Detection Using Machine Learning Methods	ICIICS (IEEE), 2023	Kaggle phishing.csv; 11,054 records, 31 attributes; 80:20 split (as reported)	Pre-extracted address-bar, HTML/JavaScript and domain attributes	SVM, RF, DT, LR, KNN, GB, AdaBoost; SHAP; Streamlit deployment	RF accuracy 0.973, DT 0.966, SVC 0.953, GB 0.946; best model deployed as a web app	Single static dataset; the reported SHAP ranking lists the class label as top attribute, so the explanation analysis needs re-examination
4	M. Boyapati, R. Aygun [10]	Phishing Web Page Detection using Web Scraping	IEEE SoutheastCon, 2023	PhishTank/OpenPhish with top-ranked benign sites (12,564 records); Kaggle (11,482); Phishpedia data	Web-scraped hybrid: URL, network, domain, content and search-engine features	XGBoost, MLP, LR, SVM, RF, DT, NB, K-means, autoencoder; with and without PCA	XGBoost best, test accuracy 0.981–0.991 on three datasets; precision/recall 98.91%/97.94% on Phishpedia data	Needs third-party/search-engine features and live scraping (latency); no visual features
5	M. Abutaha, M. Ababneh, K. Mahmoud, S. A.-H. Baddar [13]	URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis	IEEE ICICS, 2021	Public labelled corpus of 1,056,937 URLs, balanced by random under-sampling	22 lexical URL features, reduced by variance and Spearman multicollinearity filtering	SVM, gradient boosting, RF, neural network; 60:40 split	SVM best, accuracy 99.89%; neural network about 97%	URL-only evidence; well-formed URLs may evade; benign data from one compiled source
6	O. K. Sahingoz, E. Buber, O. Demir, B. Diri [8]	Machine Learning Based Phishing Detection from URLs	Expert Syst. Appl., vol. 117, 2019	Authors' own collected phishing and legitimate URL set	NLP-based word features, word vectors and hybrid features	Several classifiers; RF reported best	RF with NLP-based features reported best, about 97.98% accuracy	Language-dependent word features; no page-content evidence
7	K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, W. K. Tiong [22]	A New Hybrid Ensemble Feature Selection Framework for ML-Based Phishing Detection System	Inf. Sci., vol. 484, 2019	Authors' collections of phishing and legitimate webpages	URL and page-content feature sets ranked by the HEFS procedure	Hybrid ensemble feature selection (CDF-gradient plus data perturbation) with classifiers	RF reported as best classifier on the selected feature subset	Benefit depends on feature quality; content features require page fetching

S. No.	Author(s)	Paper Title	Publication / Year	Dataset / Data Source	Features Used	Machine Learning Method	Reported Findings	Limitations
8	S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, I. H. Sarker [11]	Modeling Hybrid Feature-Based Phishing Websites Detection Using ML Techniques	Ann. Data Sci., 2022	Public phishing/legitimate URL data with extracted URL and hyperlink features	URL-based and hyperlink-based features, and their combination	Supervised ML classifiers on individual and hybrid feature sets	URL-only and hyperlink-only sets individually insufficient; hybrid set more effective	Hyperlink features need page fetching; generalisation to new campaigns not examined
9	Y. Lin et al. [21]	Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages	USENIX Security Symp., 2021	Authors' large corpus of phishing and benign pages with screenshots	Screenshot logos and brand identity	Deep logo detection with Siamese matching	Strong precision in identifying phishing pages by brand-domain mismatch	Requires rendering and a reference brand list; heavier than URL-only inference
10	J. Ma, L. K. Saul, S. Savage, G. M. Voelker [7]	Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs	ACM SIGKDD, 2009	Labelled URL feeds from blacklists and legitimate sources	Lexical and host-based (DNS/WHOIS) features	Linear classifiers, including online learning	Learned URL classifiers flag malicious sites absent from blacklists	Host lookups add latency; features drift over time

Note: “Reported Findings” are results stated by the original authors on their own datasets; they are literature results, are not comparable across rows, and are not results of the proposed system.

4. PROPOSED AI-BASED PHISHING DETECTION FRAMEWORK

The proposed Real-Time ML-Based Phishing Detection Framework turns a URL into a verdict, a confidence value and a short explanation within a bounded time. It is made up of 10 stages.

1) **Input from URL:** User input or browser extension/API client provides a URL.
2) **Data validation:** syntax, scheme, length and encoding checks - discard incorrect data and prevent injection at later stages.
3) **URL preprocessing:** normalisation (lower casing host, decoding percent-encoding, removing default ports, removing fragments) as well as resolution of known shortener patterns.
4) **Feature extraction:** hierarchical extraction of lexical (Tier 0), domain (Tier 1) and webpage (Tier 2) features, where lower tiers are always extracted and higher tiers are only extracted, if possible, within a time limit.
5) **Feature selection:** a predetermined and proven subset lowers dimensionality and latency.
6) **ML classification:** an estimated value of p(x) is obtained by a trained classifier (or an ensemble of classifiers).
7) **Prediction confidence:** probabilities are adjusted so that the reported confidence matches what is seen in the data.
8) **Explanation:** The features which play the most important roles in the decision are identified, e.g., using SHAP-style attributions [26] or LIME [27].
9) **Final output:** the label and the confidence are returned. To display verdict as a real-time alert.
10) **Optional integration:** a REST API and browser extension are available.

The output has the form shown below. *The value 94.2% is only an example of the interface format and is not an experimental result.*

Prediction: PHISHING | Confidence: Phishing — 94.2% *(illustrative example)*

5. FEATURE ENGINEERING

5.1 Lexical URL Features

These features are computed from the URL string alone and require no network access: URL length; hostname length; number of dots, hyphens, slashes, special characters and digits; number of sub-domains; presence of an IP address as host; presence of suspicious keywords (such as “login”, “verify”, “secure”, “account”); URL entropy; character frequency; and token frequency after splitting on delimiters. Entropy is computed over the character distribution of the URL:

$$H(u) = -\Sigma c\ p(c)\ \log_2 p(c) \qquad (2)$$

where $p(c)$ is the relative frequency of character c in u. High entropy can indicate machine-generated or obfuscated strings.

5.2 Domain-Based Features

Where available, domain age, registration information, WHOIS-related fields, DNS characteristics, SSL/TLS-related indicators and domain reputation are informative; very recently registered domains are a common phishing signal [10]. It is important to separate **features extractable directly from the URL** (all of Section 5.1) from **features requiring external services** (WHOIS, DNS resolution, certificate inspection, reputation or search-engine lookups). The latter depend on third-party availability, add network delay and may reveal queried URLs to external parties, so the framework treats them as optional Tier 1 features.

5.3 Webpage Features

If page analysis is enabled (Tier 2), candidate features are the number of forms, presence of login or password fields, number of hyperlinks and the proportion pointing to external domains, number of iframes, amount and type of JavaScript, redirect behaviour, HTML/DOM statistics, and references to external resources such as images and scripts. Pages must be fetched in an isolated environment with strict timeouts, because the analyser itself may be targeted.

6. METHODOLOGY

6.1 Dataset Selection

Suitable public sources include phishing feeds such as PhishTank and OpenPhish [5] for the positive class, and a reputable popularity ranking for the legitimate class. Research datasets may also be used, including the UCI-style phishing-website feature data introduced by Mohammad *et al.* [28], the lexical URL corpus used by Abutaha *et al.* [13] and the datasets used in the studies reviewed in Section 3. Kaggle datasets will be used only where provenance and licensing are clearly documented.

Dataset used in the preliminary benchmark. The benchmark uses the public Phishing Websites dataset introduced by Mohammad *et al.* [28], retrieved programmatically from the UCI Machine Learning Repository (dataset ID 327) with the *ucimlrepo* package. It contains 11,055 instances, each described by 30 pre-computed attributes and a binary label (−1 phishing, +1 legitimate): 4,898 phishing (44.3%) and 6,157 legitimate (55.7%). According to the accompanying feature documentation, the attributes are address-bar features (e.g., *having_IP_Address*, *URL_Length*, *SSLfinal_State*), abnormal-request features (*Request_URL*, *URL_of_Anchor*), HTML/JavaScript features (*on_mouseover*, *Iframe*) and domain features (*age_of_domain*, *web_traffic*, *Page_Rank*). Raw URLs and page content are not included. A stratified 80:20 train/test split with random seed 42 gives 8,844 training and 2,211 test samples (980 phishing, 1,231 legitimate). Several attributes (domain age, web traffic, page rank) require external services, so the benchmark reflects detection quality rather than end-to-end latency. Only 5,849 of the 11,055 rows are distinct, and some identical feature vectors carry both labels; the consequences are discussed in Sections 10 and 13.

6.2 Dataset Preparation

Data cleaning, removal of data duplicates, verification of the labels (e.g., cross-checking phishing labels with more than one feed), data imputation, treatment of class imbalance (class weights or re-sampling of training data only), and a stratified train-validation-test split will be performed before training. The preliminary benchmark used a single stratified 80:20 train/test split (seed 42) without a separate validation set and without duplicate removal: 1,428 of the 2,211 test feature vectors (64.6%) also occur in the training set. A 70:15:15 split with cross-validation and deduplication is planned for the final evaluation (configuration to be finalized). Duplicates will be removed at the level of normalised URL, columns based on the label will be excluded, and a time-ordered split will be tested as well as a random split, to approximate performance on subsequent attacks.

6.3 Preprocessing

Preprocessing comprises URL normalisation, duplicate removal, tokenisation of the URL into words and sub-words where text features are used, encoding of categorical variables, feature scaling for scale-sensitive models (LR, SVM, KNN) and explicit missing-value handling, including indicator variables for unavailable Tier 1 and Tier 2 features. Scalers and encoders are fitted on training data only.

6.4 Feature Extraction

Lexical features are computed with the Python standard library and URL-parsing utilities; domain features through DNS/WHOIS/TLS queries; and webpage features by parsing fetched HTML. The tiered process is illustrated in Fig. 3 (Section 7). Every extractor will record its own execution time so that the latency contribution of each tier can be reported.

6.5 Feature Selection

Candidate methods are correlation analysis (as used by Abutaha *et al.* to remove multicollinearity [13]), information gain, chi-square tests, recursive feature elimination, embedded selection (L1-regularised models) and tree-based feature importance. Selection is performed inside cross-validation folds to avoid optimistic bias, and the final subset is chosen by validation performance and extraction cost.

6.6 Model Development

The candidate classifiers are naive Bayes, logistic regression, decision tree, random forest [14], SVM [15], gradient boosting [16] and XGBoost [17]. In the preliminary benchmark, extra trees, K-nearest neighbours, AdaBoost and a histogram-based gradient-boosting model were also evaluated. XGBoost, a character-level CNN [29] and an LSTM are planned as further baselines and have not yet been evaluated. As a proposed configuration, an ensemble of the best-performing diverse models will be combined by soft voting or stacking, with hyperparameters tuned by stratified cross-validation and probabilities calibrated on the validation set.

6.7 Real-Time Deployment

The final model could be served through a FastAPI or Flask REST endpoint that accepts a URL and returns a JSON object with label, confidence and top features, and consumed by a browser extension or web application. **Deployment is a proposed design only; no deployed system is claimed.** A local-inference option and a server-side option will be compared with respect to latency and URL privacy.

7. PROPOSED SYSTEM ARCHITECTURE

Fig. 1 shows the proposed architecture. A request from the user or browser passes through validation and preprocessing, parallel URL and webpage feature extraction, feature selection and the ML model; the prediction is accompanied by a confidence score and explanation and returned as an alert or API response. Fig. 2 shows the offline training and online inference workflows, and Fig. 3 illustrates the tiered feature-extraction pipeline with an invented example URL.

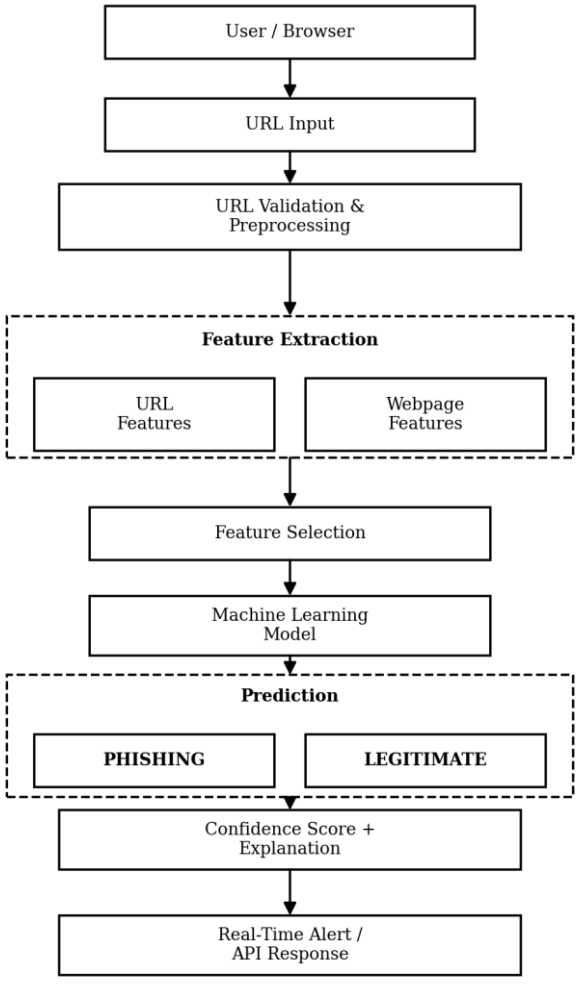


Fig. 1. Proposed Real-Time Phishing Website Detection Architecture.

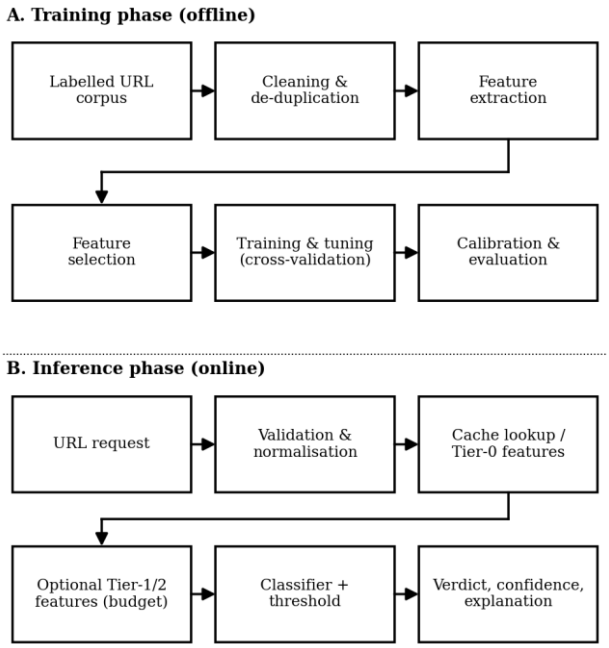


Fig. 2. Phishing Detection Workflow (offline training and online inference).

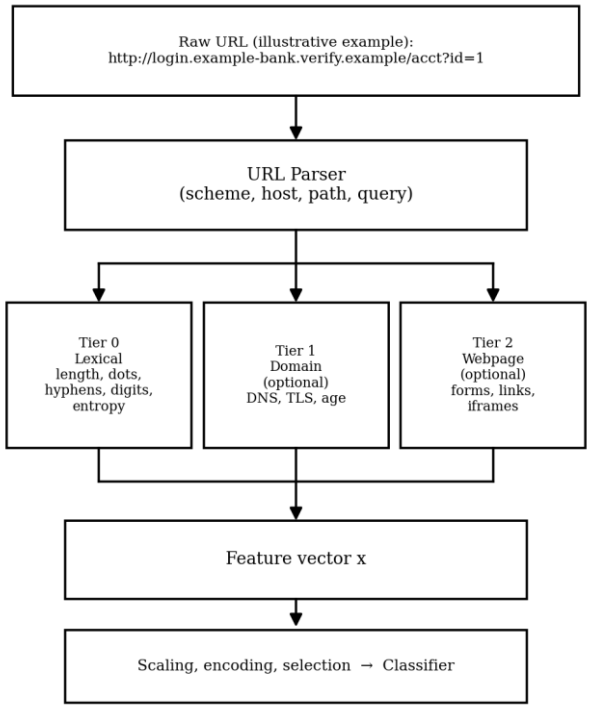


Fig. 3. Example Feature Extraction Pipeline (illustrative URL; tiers 1 and 2 optional).

8. PROPOSED ALGORITHM

Algorithm 1: Real-Time Phishing Website Detection
Input: URL U; trained classifier M; feature selector S; threshold τ; time budget B
Output: class C, confidence P, explanation E

receive U
if not VALID(U) **then return** ERROR
$U' \leftarrow$ NORMALIZE(U)
if U' in CACHE **then return** CACHE[U']
$x \leftarrow$ EXTRACT_LEXICAL(U')
if budget B remains **then** $x \leftarrow x \cup$ EXTRACT_DOMAIN_PAGE(U', B)
$x \leftarrow$ SCALE_ENCODE($S(x)$), flagging unavailable features
$p \leftarrow M$.PREDICT_PROBA(x) for the phishing class
if $p \geq \tau$ **then** $C \leftarrow$ PHISHING, $P \leftarrow p$
else $C \leftarrow$ LEGITIMATE, $P \leftarrow 1 - p$
$E \leftarrow$ TOP_K_CONTRIBUTING_FEATURES(M, x)
store (U', C, P, E) in CACHE
return C, P, E

Step 6 realises the latency policy: Tier 1 and Tier 2 features are requested only while time remains, and a missing-feature indicator lets the classifier handle partial evidence.

9. EXPERIMENTAL SETUP

Environment of the preliminary benchmark. The benchmark was run on a laptop with an Intel Core Ultra 5 125H processor (14 cores: 4 performance, 8 efficient and 2 low-power efficient; 18 threads), 16 GB LPDDR5 RAM, integrated Intel Arc graphics and Windows 11. The software was Python 3.11 with pandas 2.2.3, NumPy 1.26.4, Matplotlib 3.9.2, ucimlrepo 0.0.7, joblib 1.4.2 and scikit-learn 1.9.1 [30]. A single random seed of 42 was used for the train/test split and for every model with a random component. All models were scikit-learn pipelines with median imputation; naive Bayes, logistic regression, SVM and KNN additionally used standardisation. Random forest and extra trees used 250 trees, AdaBoost 120 estimators, KNN five neighbours and the SVM an RBF kernel; logistic regression, decision tree, random forest, extra trees and SVM used balanced class weights, and all other settings were library defaults. No hyperparameter tuning was performed. Random forest, extra trees and KNN used all available CPU threads. Prediction time is the wall-clock time of one batch prediction over the 2,211 test samples divided by the number of samples, on pre-computed features, and excludes feature extraction. XGBoost [17] is included as an optional step in the benchmark script but is not part of the recorded results; Jupyter Notebook or Google Colab and FastAPI or Flask are planned tools not yet used. In the final evaluation, prediction time will be measured per URL as the median and 95th percentile over repeated requests, with feature-extraction and model-inference time reported separately, with and without caching.

10. EXPERIMENTAL RESULTS AND DISCUSSION

Table II reports measured results of the preliminary benchmark for ten classical classifiers on one held-out test set (2,211 samples). Models that have not yet been evaluated (XGBoost, a character-level CNN, an LSTM and the proposed ensemble) are not listed; they are deferred to Section 16.

TABLE II
MODEL PERFORMANCE COMPARISON

Model	Acc.	Prec.	Rec.	F1	AUC	Pred. Time (ms/sample)
Naive Bayes	0.6119	0.5332	**0.9990**	0.6953	0.9677	0.0019
Logistic Regression	0.9317	0.9295	0.9153	0.9224	0.9786	0.0019
SVM (RBF)	0.9566	0.9576	0.9439	0.9507	0.9901	0.1637
Decision Tree	0.9701	0.9741	0.9582	0.9660	0.9772	**0.0014**
Random Forest	0.9765	**0.9793**	0.9673	0.9733	**0.9967**	0.0398
Extra Trees	**0.9774**	0.9784	0.9704	**0.9744**	0.9948	0.0403
Gradient Boosting	0.9502	0.9494	0.9378	0.9435	0.9913	0.0034
Hist. Gradient Boosting	0.9688	0.9710	0.9582	0.9646	0.9957	0.0059
AdaBoost	0.9322	0.9415	0.9031	0.9219	0.9867	0.0212
KNN	0.9471	0.9481	0.9316	0.9398	0.9856	0.0586

Acc. = accuracy, Prec. = precision, Rec. = recall, F1 = F1-score, AUC = ROC-AUC. Prediction time is mean batch-inference time per sample on pre-computed test features, excluding feature extraction. Bold = best value in the column. Single held-out split; no repeated runs or confidence intervals yet.

Metrics will be interpreted as follows, where TP, TN, FP and FN are defined in Section 11.

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (3)$$

$$Precision = TP / (TP + FP) \quad (4)$$

$$Recall = TP / (TP + FN) \quad (5)$$

$$F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall) \quad (6)$$

Overall percentage of accurate predictions **(accuracy)** may be misleading if classes are imbalanced. **Precision** is the percentage of the websites that are predicted to be phishing that actually are. **Recall** is the percentage of the actual phishing websites that are properly identified. Precision and Recall are combined as **F1-Score**. **ROC-AUC** reflects the capability of the classifier in distinguishing phishing from legitimate websites at all thresholds. **Prediction time** is a measure of suitability for real time use. For the final evaluation, the results will be presented as mean and standard deviation over the folds of the cross-validation and paired significance tests will be conducted if comparing models.

Initial observations on the initial benchmark. Among the competitive models, tree ensembles achieved the best results with Extra Trees having the highest accuracy (0.9774) and the highest F1-score (0.9744), whereas Random Forest had the highest precision (0.9793) and the highest ROC-AUC (0.9967). The two are tied on two tests, total errors (50 vs 52), and as such, this split is inconclusive. Linear and instance based models performed poorer (logistic regression 0.9317 accuracy, KNN accuracy 0.9471). Naive Bayes achieved the best recall of 0.9990, but by misclassifying a majority of valid URLs as phishing, its accuracy of 0.6119 and precision of 0.5332, and ROC-AUC value of 0.9677, shows its ranking is useful, but that the default decision threshold for Naive Bayes is not well suited to this data. The mean batch inference time ranged from 0.0014ms (decision tree) to 0.1637ms (RBF-SVM) and was around 0.04ms for Extra Trees and Random Forest, which is not end-to-end latency but only the inference time.

Supplementary check on duplicates (Extra Trees only). Table II is likely optimistic, because only 5,849 of the 11,055 rows are distinct and 64.6% of test vectors also appear in the training set. Repeating the Extra Trees pipeline with the same settings after removing duplicate rows (5,849 rows: 3,019 phishing, 2,830 legitimate; stratified 80:20 split with seed 42, giving 4,679 training and 1,170 test samples) gave accuracy 0.9410, precision 0.9573, recall 0.9272, F1-score 0.9420 and ROC-AUC 0.9579 (TN 541, FP 25, FN 44, TP 560), with only 24 test vectors shared with the training set. The other models were not re-run, so Table II remains the benchmark on the original, duplicate-containing data.

11. CONFUSION MATRIX

TABLE III
CONFUSION MATRIX OF THE BEST MODEL (EXTRA TREES)

	Predicted Legitimate (0)	**Predicted Phishing (1)**
Actual Legitimate (0)	TN = 1210	FP = 21
Actual Phishing (1)	FN = 29	TP = 951

Measured counts on the held-out test set (2,211 samples: 1,231 legitimate, 980 phishing).

TP: phishing correctly detected. **TN:** legitimate website correctly identified. **FP:** legitimate website incorrectly classified as phishing. **FN:** phishing website incorrectly classified as legitimate. For Extra Trees, FPR = FP/(FP + TN) = 21/1231 = 1.71% and the false-negative rate FN/(FN + TP) = 29/980 = 2.96%. Table III-B lists the counts of all evaluated models.

TABLE III-B
CONFUSION-MATRIX COUNTS OF EVALUATED MODELS

Model	**TN**	**FP**	**FN**	**TP**
Naive Bayes	374	857	1	979
Logistic Regression	1163	68	83	897
SVM (RBF)	1190	41	55	925
Decision Tree	1206	25	41	939
Random Forest	1211	20	32	948
Extra Trees	1210	21	29	951
Gradient Boosting	1182	49	61	919
Hist. Gradient Boosting	1203	28	41	939
AdaBoost	1176	55	95	885
KNN	1181	50	67	913

Measured counts, single held-out test set.

For all nine models other than naive Bayes, false negatives exceeded false positives (for Extra Trees 29 versus 21; for AdaBoost 95 versus 55), meaning that missed phishing sites were more common than false alarms at the default threshold. Because missed phishing is costly, lowering the threshold τ should be examined as a way to raise recall at some cost in false positives. Naive Bayes shows the opposite pattern (857 false positives, 1 false negative).

12. BASELINE VS PROPOSED MODEL

A qualitative comparison of the approaches is given in Table IV, based on general properties of each model family. These are expectations, not measured results; here no performance numbers are given and measured values for the evaluated models are in Table II.

TABLE IV
BASELINE AND PROPOSED APPROACH COMPARISON

Method	Feature Type	Training Complexity	Inference Speed	Expected Strength	Expected Limitation	Real-Time Suitability
Naive Bayes	Lexical / tabular	Very low	Very fast	Simple, fast baseline	Independence assumption	High
Logistic Regression	Lexical / tabular	Low	Very fast	Interpretable coefficients	Linear decision boundary	High
SVM	Scaled lexical / tabular	Moderate to high (kernel)	Fast (linear) to moderate (kernel) [15]	Effective in high-dimensional spaces	Needs scaling; kernel cost grows with data	High (linear), moderate (kernel)
Random Forest	Lexical + optional domain/page	Moderate	Fast to moderate (many trees)	Robust to noise; handles non-linearity [14]	Larger memory footprint	High
XGBoost	Lexical + optional domain/page	Moderate	Fast	Strong on tabular data [17]	Sensitive to tuning	High
CNN	Character-level URL sequences	High	Moderate	Learns patterns without manual features [29]	Needs more data; harder to interpret	Moderate
LSTM	Character / token sequences	High	Moderate to slow	Models order dependence	Sequential computation	Moderate to low
Proposed Hybrid / Ensemble Model	Tiered lexical + domain + webpage	Moderate to high (ensemble + calibration)	Tier-dependent: fast at Tier 0, slower with Tier 1–2	Complementary errors; calibrated confidence and explanation	Added complexity; latency of optional tiers	To be determined by measurement

Qualitative expectations based on general properties of each family; to be verified experimentally.

13. DISCUSSION

Throughout, “literature evidence” refers to findings reported by other authors, “preliminary measurement” refers to the results in Table II, and “expectation” refers to anticipated behaviour that must still be verified.

Lexical features. *Literature:* URL-only features have supported high reported accuracy [13], [8] and need no page fetch. *Expectation:* they should supply a fast first-tier decision, but not be sufficient against well-formed URLs.

Tree ensembles and SVM. *Literature:* RF and XGBoost were reported as best in several studies [18], [10], while an SVM was best in another [13]. *Expectation:* RF and XGBoost are strong candidates because they handle mixed, non-linear tabular features; SVMs may suit high-dimensional sparse token features. The ranking must be established empirically. *Preliminary measurement:* Extra Trees and Random Forest had the highest accuracy (0.9774 and 0.9765) and the RBF-SVM was lower (0.9566), but the margin between the two leaders is two samples on a single split.

Webpage features. The combination of hybrid URL + content was reported as being more effective than either separately [11]. Cost of page features is added evidence (e.g. login forms posting to foreign hosts); cost of latency is expectation.Expectation is added cost of page features: e.g. login forms posting to foreign hosts.

Recall and accuracy–latency dichotomy. It was reported that multimodal detectors were much slower than unimodal detectors [9]. Of course, recall is important for security: if a user goes to a phishing site, the user is exposed, but if the user is denied access for a false hit, then his trust is lost; the threshold τ will be determined by the precision–recall curve within the latency budget. Preliminary measurement: For all ten models, the mean batch-inference time per sample is less than 0.2 ms, except for feature extraction which is expected to dominate if features from the domain or page are not inputted.

Class imbalance and dataset drift. However, since real traffic is mostly legitimate URLs, the accuracy may be hiding poor recall, Abutaha et al. under-sampled [13] to balance an extremely imbalanced corpus. Phishing attacks also evolve across time and transient sites make data scarce [9] so time-ordered evaluation and regular retraining is necessary to make generalisations in the light of new attacks. The preliminary test set is slightly imbalanced (1,231 is a legitimate website and 980 are phishing sites).

Restrictions on the preliminary results. They come from a single held-out split, without repeated runs, cross-validation, a time-ordered split or adversarial testing, and use pre-computed features. The dataset contains many duplicate rows (5,849 distinct of 11,055), which inflates the scores in Table II (see Section 10); in addition, results from a static data set might overestimate performance on new campaigns.

Adversarial robustness and deployment. In the case of URL-only detectors [9] it is reported that they are not robust enough and need to be tested, such as by adversarial retraining [24]. Some deployment issues are: fetch safety, timeouts, caching, model updates and URL privacy.

14. ETHICAL CONSIDERATIONS AND LIMITATIONS

14.1 Dataset Bias

Training data may not represent all phishing campaigns, languages or geographic regions, and feed-derived samples favour attacks that have been reported.

14.2 False Positives

Incorrectly flagging legitimate websites can disrupt users and harm businesses; the threshold and an appeal or override path need careful design.

14.3 False Negatives

Missing a phishing website can expose users to credential theft and financial loss; the system must be positioned as one layer of defence, not a guarantee.

14.4 Dataset Quality

Incorrect labels, duplicate URLs and outdated samples can inflate or distort measured performance.

14.5 Adversarial Attacks

Attackers may deliberately manipulate URLs or pages to evade ML models [23], [25]; adversarial samples should be generated only in controlled research environments.

14.6 Privacy

Submitting URLs, and domain or page data, to a server may reveal browsing activity or sensitive tokens in query strings; local inference, data minimisation and retention limits are advisable.

14.7 Explainability

Users and analysts should receive understandable reasons for a verdict, using attribution methods [26], [27], with the caveat that explanations can themselves be misleading if data leakage exists.

14.8 Multilingual and Cross-Domain Limitations

Models trained on particular datasets may not generalise equally across languages, domains and countries; language-dependent word features are especially affected [8].

15. CONCLUSION

However, phishing is still a significant cybersecurity issue and traditional blacklists don't work well when it comes to new and modified variants because they are reactive. The discriminative pattern learning of machine learning from URL and webpage features has been shown to be useful in previous studies, but with significant differences in the results depending on the datasets and protocols. There are trade-offs between accuracy, robustness and computational cost for different algorithms. The proposed Real-Time ML-Based Phishing Detection Framework is designed to deliver low-latency, accurate detection, with confidence rating and explanations. Tree-based ensembles outperformed other classical classifiers in a preliminary evaluation on a held-out test set with an accuracy of 0.9774 and an F1-score of 0.9744, with the batch inference time of trees at about 0.04 ms per sample, which was considerably shorter than that of naive Bayes at its default threshold. These are the results of one split of data containing many duplicate rows, and do not include feature-extraction time. The full framework, the deep-learning baselines and the proposed ensemble have yet to be implemented and evaluated, and no performance is claimed for them.

16. FUTURE WORK

Immediate next steps are to train and evaluate the models not yet benchmarked (XGBoost, a character-level CNN, an LSTM and the proposed ensemble), add them to the results table, and measure end-to-end per-URL latency of the real-time framework. After implementation and evaluation, planned extensions include browser-extension integration; a real-time API; continuous or online learning to follow concept drift; multilingual phishing detection; multimodal and visual webpage analysis, building on logo-based approaches [21]; domain-reputation integration; adversarial training [24]; stronger explainable-AI support; federated or privacy-preserving learning; integration with threat-intelligence platforms; and integration with automated fact and security verification systems.

REFERENCES

[1] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, “Phishing attacks: A recent comprehensive study and a new anatomy,” *Front. Comput. Sci.*, vol. 3, Art. no. 563060, Mar. 2021.

[2] M. Khonji, Y. Iraqi, and A. Jones, “Phishing detection: A literature survey,” *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2091–2121, 4th Quart. 2013.

[3] Anti-Phishing Working Group, “Phishing activity trends reports,” APWG. [Online]. Available: https://apwg.org/trendsreports/

[4] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "PhishNet: Predictive blacklisting to detect phishing attacks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.
[5] S. Bell and P. Komisarczuk, "An analysis of phishing blacklists: Google Safe Browsing, OpenPhish, and PhishTank," in *Proc. Australasian Computer Science Week Multiconf. (ACSW)*, 2020, Art. no. 3, doi: 10.1145/3373017.3373020.
[6] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, "PhishTime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in *Proc. 29th USENIX Security Symp.*, 2020, pp. 379–396.
[7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2009, pp. 1245–1254.
[8] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, Mar. 2019.
[9] P. T. Duy, V. Q. Minh, B. T. H. Dang, N. D. H. Son, N. H. Quyen, and V.-H. Pham, "A study on adversarial sample resistance and defense mechanism for multimodal learning-based phishing website detection," *IEEE Access*, vol. 12, pp. 137805–137824, 2024, doi: 10.1109/ACCESS.2024.3436812.
[10] M. Boyapati and R. Aygun, "Phishing web page detection using web scraping," in *Proc. IEEE SoutheastCon*, 2023, pp. 167–174, doi: 10.1109/SoutheastCon51012.2023.10115148.
[11] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Ann. Data Sci.*, pp. 1–26, 2022.
[12] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 639–648.
[13] M. Abutaha, M. Ababneh, K. Mahmoud, and S. A.-H. Baddar, "URL phishing detection using machine learning techniques based on URLs lexical analysis," in *Proc. 12th Int. Conf. Information and Communication Systems (ICICS)*, 2021, pp. 147–152, doi: 10.1109/ICICS52457.2021.9464539.
[14] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[16] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.
[17] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[18] S. Anakal, K. Maka, A. Tadkal, S. Humanabad, S. Anakal, and L. E, "Phishing website detection using machine learning methods," in *Proc. Int. Conf. Integrated Intelligence and Communication Systems (ICIICS)*, 2023, doi: 10.1109/ICIICS59993.2023.10420933.
[19] A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing detection system through hybrid machine learning based on URL," *IEEE Access*, vol. 11, pp. 36805–36822, 2023, doi: 10.1109/ACCESS.2023.3252366.
[20] V. Shahrivari, M. M. Darabi, and M. Izadi, "Phishing detection using machine learning techniques," 2020, *arXiv:2009.11116*.
[21] Y. Lin, R. Liu, D. M. Divakaran, J. Y. Ng, Q. Z. Chan, Y. Lu, Y. Si, F. Zhang, and J. S. Dong, "Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages," in *Proc. 30th USENIX Security Symp.*, 2021, pp. 3793–3810.
[22] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Inf. Sci.*, vol. 484, pp. 153–166, May 2019.
[23] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Eur. Conf. Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Berlin, Germany: Springer, 2013, pp. 387–402.
[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[25] Y. Yuan, G. Apruzzese, and M. Conti, "Multi-SpacePhish: Extending the evasion-space of adversarial attacks against phishing website detectors using machine learning," *Digit. Threats: Res. Pract.*, vol. 5, no. 2, pp. 1–51, Jun. 2024.
[26] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 4765–4774.
[27] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
[28] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Comput. Appl.*, vol. 25, no. 2, pp. 443–458, Aug. 2014.
[29] E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad, and A. I. A. Alzahrani, "A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators," *Sensors*, vol. 23, no. 9, p. 4403, Apr. 2023.
[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.