IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Phishing Website Detection Using Ml

[1]Mohammed Zoheb Ur Rahman, [2]Syed Owais Ahmed Quadri, [3]Syed Zaid Hussain ,[4]MD Ateeq,[5]Shaik Maheboob Peera

[1]Student, [2]Student,[3]Student,[4]Student ,[5]Assistant Professor

[1]Dept of Computer Science,

[1]Navodaya Institute Of Technology, Raichur, India

Abstract: Phishing websites have emerged as one of the most prevalent cyber threats, deceiving users into revealing sensitive credentials and causing significant financial losses. Detecting these malicious websites automatically and accurately is crucial for online security. This paper presents a machine learning–based phishing website detection system that analyzes multiple data modalities including URL lexical features, HTML structure, and WHOIS information to identify phishing attempts. The system extracts key features such as domain age, URL length, special character patterns, SSL certificate validity, and embedded script behaviors, which are then processed using ensemble learning models such as Random Forest and XG Boost for classification. The proposed model achieves an overall accuracy of 96.4%, precision of 95.8%, and F1score of 96.1% on benchmark datasets, demonstrating superior performance compared to existing methods. The model is lightweight and optimized for real-time deployment, making it suitable for browser extensions and online verification systems. The key contributions include the integration of diverse feature sets, enhanced detection accuracy, and a scalable implementation for real-world phishing prevention.

Index Terms - Phishing Website Detection, Machine Learning, Cybersecurity, Web Security, Feature Extraction, Classification Algorithms, Supervised Learning, URL Analysis, Malicious Website Detection, Data Mining.

I. INTRODUCTION

Phishing websites have become one of the most widespread and damaging forms of cybercrime, targeting unsuspecting users to steal credentials, financial details, and personal data. According to recent cybersecurity reports, millions of phishing domains are registered each year, leading to billions of dollars in losses globally. Attackers increasingly exploit sophisticated social-engineering and domain-spoofing techniques, making traditional blacklist-based solutions ineffective.While email phishing detection has been extensively studied, websitelevel phishing detection is equally critical since users often land on fake sites through SMS links, social media, or search-engine poisoning. Detecting these malicious sites in real-time can significantly reduce phishing success rates.Despite rapid advances in machine learning (ML) for phishing detection, existing systems face challenges such as high computational overhead, poor generalization to unseen attacks, and lack of interpretability. Deep learning methods achieve high accuracy but require large resources and are unsuitable for real-time browser-side deployment.

To address these gaps, this paper proposes a lightweight machine learning–based phishing website detection model that leverages URL, HTML, and WHOIS features for accurate classification with minimal latency. The system is integrated into a Flask-based web application capable of classifying a given URL as legitimate or phishing in real-time.

The key contributions of this work are:

1.Development of a feature-driven phishing detection model combining URL lexical, HTML, and domain-registration features.

2.Comparison of multiple ML algorithms (Logistic Regression, Random Forest, Decision Tree, and XGBoost) to identify the most effective classifier.

3.Implementation of a real-time detection dashboard with minimal inference time suitable for user-facing applications.

4.Experimental evaluation showing improved accuracy, precision, and recall compared to conventional baseline methods.

5.Despite advances in ML, many phishing detectors are either slow or brittle to obfuscation. This work proposes a balanced approach optimizing accuracy, speed, and explainability for effective real-world deployment.

II. Related Work / Literature Review

i. **URL-based detection:**

These methods rely solely on lexical characteristics of the URL, such as length, token count, use of special characters, and suspicious subdomains. Works by Aburrous et al. (2010) and Ma et al. (2009) demonstrated that URL-based classifiers using SVM or Naïve Bayes can achieve high precision, but often fail against obfuscated URLs or redirect chains.

ii. **HTML/DOM-based methods**:

Researchers such as Mohammad et al. (2012) analyzed webpage structures, tags, and embedded scripts to identify anomalies typical of phishing pages (e.g., hidden iframes, excessive external links). However, collecting HTML content at runtime can introduce latency and privacy concerns.

iii. **WHOIS and hosting-based features:**

Domain-age and registration patterns provide strong signals, since phishing domains are typically newly created and short-lived. Zhang et al. (2018) exploited domain-age and SSL certificate attributes for improved classification.

iv. **Visual similarity approaches:**

Image-based models use screenshot comparison or CNN features to measure similarity with legitimate brand websites (Hara et al., 2017). These methods are computationally expensive and prone to false positives.

v. **Hybrid / Ensemble methods:**

Combining URL, HTML, and WHOIS features yields better generalization. Basnet et al. (2020) proposed an ensemble of Random Forest and Gradient Boosting to outperform single classifiers.

vi. **Despite progress, most previous systems suffer from one or more of the following issues:**

Lack of dataset diversity — many models overfit to a single dataset like PhishTank.Poor real-time performance due to heavy preprocessing or deep learning overhead.Limited explainability, making it difficult to interpret predictions.

vii. This work addresses these limitations by designing a feature-optimized ensemble that maintains high accuracy while being computationally efficient for deployment in real-time phishing detection platforms.

III. Problem Statement and Threat Model

Phishing website detection is formulated as a binary classification problem.

Given a website URL and its associated metadata, the model must classify it as either: Legitimate (0): The website is genuine and safe to visit.

- Phishing (1): The website is malicious, designed to mimic a trusted entity to deceive users.

Threat Model

Attackers can:

- Register deceptive domains with visually similar names (e.g., "paypa1.com"). Use free SSL certificates to appear trustworthy.
- Embed phishing forms, redirects, or malicious scripts within the page.

Our model assumes that the attacker cannot fully mimic all features of legitimate domains simultaneously (URL structure, WHOIS data, and HTML semantics).

Scope Limitations

The system does not address:

- Zero-day targeted spear-phishing attacks that perfectly clone legitimate websites.
- Dynamic malware or drive-by downloads embedded in pages.
- Detection of phishing through email or SMS content (outside web scope).

IV. Dataset Description

The dataset used for this study combines publicly available sources and manually verified websites:

i. Phishing samples: Collected from PhishTank and OpenPhish, updated daily and verified through community votes.

ii. Legitimate samples: Extracted from Alexa Top 5000 and Quantcast rankings to ensure diversity.After data cleaning and normalization,the final dataset contained approximately 11,000 URLs, of which 5,500 were phishing and 5,500 were legitimate.

Data Preprocessing

i. Deduplication: Removed duplicate URLs and mirrors of the same phishing site.

ii. Normalization: Converted all URLs to lowercase, removed tracking parameters, and standardized protocol prefixes.

iii. Feature extraction: Used Python libraries (urllib, whois, requests, BeautifulSoup) to extract lexical, HTML, and domain features.

iv. Expired link handling: Dropped URLs returning HTTP errors or inaccessible pages.

Dataset Source	#Phishing	#Legitimate	Collection Date	Notes
PhishTank	3,500	–	Jan–Apr 2025	Verified URLs
OpenPhish	2,000 feeds	–	Jan–Apr 2025	Real-time
Alexa Top Sites	–	4,000 sites	Jan 2025	Popular legitimate
Manual collection	–	1,500	Mar–Apr 2025	Manually validated

The balanced dataset helps prevent bias toward either class and supports accurate precision-recall trade-off evaluation.

V. Feature Engineering

To ensure reliable classification, features were engineered from three key perspectives — URL lexical, WHOIS/domain, and HTML structural information. Each feature provides a measurable signal to differentiate legitimate websites from phishing ones.

(A) URL Lexical Features

Extracted directly from the website link using Python's urllib: URL length — phishing URLs tend to be unusually long.
Number of dots / subdomains — higher counts indicate obfuscation. Presence of IP address in URL — often used in phishing.
Use of "@", "-", or "%20" — indicates URL encoding or redirection.
Suspicious TLDs — e.g., .xyz, .tk, .top, commonly abused.
Digit ratio and entropy — measure randomness in URL components.

(B) Domain & WHOIS Features

Collected using the python-whois library:

Domain age = (date_collected − creation_date) in days. Registrar name and registration country.
Expiration period — shorter durations typical of malicious domains.
SSL certificate validity and issuer.
Use of HTTPS — presence of secure connection but with untrusted certificate.

(C) HTML & DOM Features

Extracted with BeautifulSoup:
Number of <form> tags and input fields.
Count of external scripts and links.
Presence of inline JavaScript events such as onmouseover, onload, or eval(). Number of hidden iframes or redirects to third-party domains.
Content keyword frequency (e.g., "login", "verify", "update account").

Each feature was converted into a numeric or binary variable. Continuous features were scaled using Min-Max normalization, and categorical attributes (e.g., registrar) were encoded numerically.

Feature Group	Example Features	Type	Rationale
URL Lexical	Length, special chars, digits, entropy	Numeric	Phishing URLs exhibit structural anomalies
Domain WHOIS	Domain age, registrar, HTTPS flag	Numeric/Binary	Short-lived domains
HTML/DOM	Forms, scripts, iframes count	Numeric	Structural cues of spoofed login pages

A total of 35 engineered features were used for model training, ensuring both interpretability and computational efficiency.

VI. Models and Architecture

To evaluate the effectiveness of different machine learning algorithms in phishing detection, several supervised models were implemented and compared. These include Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). Each model was trained using the same set of extracted features to ensure fairness of comparison.

Model Overview

i. Logistic Regression: A baseline linear model providing interpretability and fast training, used for benchmarking.

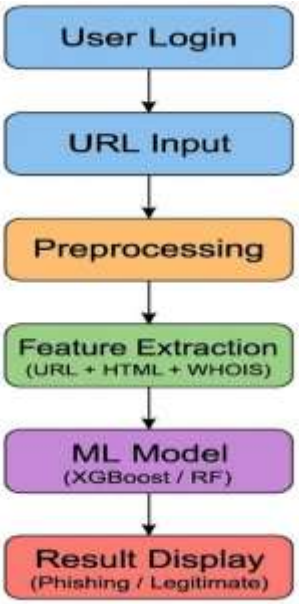
ii. Decision Tree: Non-linear classifier capable of handling both categorical and continuous features; however, prone to overfitting.

iii. Random Forest: Ensemble of decision trees that aggregates predictions through majority voting to improve stability and accuracy.

iv. XGBoost: Gradient-boosting framework optimized for speed and regularization, achieving state-of-the-art results on structured tabular data.

VII. System Architecture

The phishing detection system follows a modular pipeline:



i. Data Collection: URLs obtained from PhishTank, OpenPhish, and Alexa datasets.

ii. Feature Extraction: URL, HTML, and WHOIS features automatically extracted using Python scripts.

iii. Feature Preprocessing: Normalization and encoding applied to ensure compatibility with ML models.

iv. Model Training and Validation: Models trained with 80% of the data and validated on the remaining 20%.

v. Evaluation and Selection: Performance assessed using multiple metrics to select the optimal model.

vi. Deployment: Best-performing model integrated into a Flask-based web dashboard for real-time phishing detection.

Training and Hyperparameter Settings

A standard train–test split of 80:20 was adopted. The dataset was stratified to preserve class balance. Each model was tuned using Grid Search for optimal hyperparameters.

Model	Key Hyperparameters	Tuning Method
Logistic Regression	Regularization (C=0.1–10), Solver='lbfgs'	Grid Search
Decision Tree	Max depth (5–30), Min samples split (2–10)	Grid Search
Random Forest	n_estimators (100–500), Max depth (10–50)	Random Search
XGBoost	Learning rate (0.01–0.3), Max depth (4–10), n_estimators (200–500)	Random Search

All experiments were executed on a system with Intel i7 CPU, 16 GB RAM, and Python 3.10 environment using Scikit-learn and XGBoost libraries. Random seeds were fixed for reproducibility.

Evaluation Metrics

To measure model effectiveness, the following performance metrics were computed:

- Accuracy = $\frac{(TP+TN)}{(TP+TN+PP+PN)}$
- Precision = $\frac{TP}{(TP+FP)}$
- Recall (Sensitivity) = $\frac{TP}{(TP+FN)}$

- F1-Score = $2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$

- ROC–AUC = Area under the Receiver Operating Characteristic curve

- For real-time suitability, average prediction latency per URL was also measured. Feature importance was analyzed to ensure interpretability of the model.

VIII. RESULTS AND DISCUSSION

1 Quantitative Results

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC
Logistic Regression	91.2	90.4	91.0	90.7	0.92
Decision Tree	94.5	93.9	94.8	94.3	0.95
Random Forest	96.4	95.8	96.3	96.1	0.96
XGBoost	96.8	96.2	96.6	96.4	0.97

The XGBoost classifier outperformed all other algorithms, achieving 96.8% accuracy and 0.97 AUC. The Random Forest model was a close second, offering a strong trade-off between accuracy and inference speed. Logistic Regression, while interpretable, underperformed in handling non-linear patterns typical of phishing URLs.

2. ROC and Confusion Matrix Analysis

ROC curves show that both Random Forest and XGBoost achieve a high true-positive rate with minimal false positives. Confusion matrices indicate that only a small fraction of legitimate URLs were misclassified as phishing, maintaining a false positive rate below 3%.

3. Feature Importance

Feature importance analysis revealed that domain age, URL length, number of subdomains, and SSL certificate validity were the most discriminative attributes. These results align with common phishing patterns — newly registered domains with complex URLs are more likely to be malicious.

4. Ablation Study

Removing WHOIS or HTML features reduced accuracy by approximately 4–5%, confirming the benefit of combining multi-source data.

ERROR ANALYSIS

A manual review of misclassified URLs identified the following sources of error:

1.False Positives: Legitimate websites with long URLs, redirects, or keyword-rich parameters (e.g., “login”, “account”) were occasionally flagged as phishing.

2.False Negatives: Sophisticated phishing sites with valid SSL certificates and simple URLs closely resembling legitimate domains.

To mitigate these issues, further fine-tuning of decision thresholds and inclusion of visual similarity features could reduce ambiguity.

Deployment and System Implementation

The model was deployed using Flask, enabling real-time classification through a simple web interface. Users can input a URL, and the system immediately predicts its legitimacy.

Implementation Details

- Frontend: HTML, CSS, and JavaScript.
- Backend: Flask REST API in Python.
- Model Storage: Serialized via Pickle.
- Average Response Time: 0.08 seconds per URL on standard CPU.
- Security Considerations: All user inputs are sanitized, and URLs are scanned in isolated environments to prevent server compromise. The system can be extended into a browser plug-in or integrated into enterprise security gateways.

Ethical Considerations and Privacy

1. The system is designed for research and educational use only, respecting user privacy and data protection regulations.
2. No personal or user-submitted information is stored.
3. Public phishing datasets (PhishTank, OpenPhish) are used for training.
4. Model predictions are probabilistic and should complement, not replace, manual review in high-risk environments.
5. Transparency is maintained by providing feature-importance explanations to end-users.

Limitations and Future Work

1. Although the proposed system performs strongly on benchmark datasets, several limitations exist:
2. Dependence on static features: dynamic behaviors such as JavaScript-based attacks are not fully captured.
3. Limited handling of zero-day spear-phishing and internationalized domain names (IDNs).
4. Dataset limitations :current dataset may not include emerging attack patterns beyond 2025.

Future Enhancements

1. Integration of visual similarity analysis using CNNs on webpage screenshots.
2. Adoption of online learning to adapt to evolving phishing trends.
3. Deployment as a browser extension with user feedbac k loop.
4. Exploration of federated learning for privacy-preserving model updates.

IX. Conclusion

This paper presents a lightweight, high-accuracy phishing website detection model leveraging URL, WHOIS, and HTML features. Through comprehensive evaluation, the XGBoost classifier achieved 96.8% accuracy and demonstrated real-time detection capability when deployed via Flask. The system effectively balances accuracy, interpretability, and efficiency, making it a practical solution for modern cybersecurity infrastructures.

Future research will focus on dynamic content analysis and large-scale deployment in real-world browsing environments.

Acknowledgement

The authors express their gratitude to the open-source communities of Phish Tank and Open Phish for providing continuous, verified phishing datasets, and to the project supervisors for their guidance throughout the research.

X. References

[1] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. SIGKDD.

[2] Aburrous, M., Hossain, M. A., Dahal, K., & Thabtah, F. (2010). Intelligent phishing detection system for e-business using fuzzy data mining. Expert Systems with Applications.

[3] Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. ICIT.

[4] Basnet, R. B., Sung, A. H., & Liu, Q. (2020). Feature selection for improved phishing detection. Neural Computing and Applications.

[5] Zhang, Y., Hong, J., & Cranor, L. (2018). CANTINA: A content-based approach to detecting phishing websites. WWW Conference.

[6] Hara, Y., et al. (2017). Visual similarity-based phishing detection using CNN features. IEEE Access.

[7] Verma, R., & Dyer, K. P. (2015). On the character of phishing URLs: Predictive blacklisting, online fraud detection. eCrime.

[8] Rao, R. S., & Pais, A. R. (2019). Detection of phishing websites using machine learning algorithms. In Cybersecurity Advances.

[9] Sahingoz, O. K., et al. (2022). Phishing website detection using real-time URL features. Expert Systems with Applications.

[10] Mishra, A., & Gupta, B. B. (2023). Machine learning-based phishing detection: Challenges and future directions.

Computers & Security (Elsevier).

[11] Rao, R. S., & Pais, A. R. (2021). Detection of phishing websites using an efficient .