Journal of Advanced Artificial Intelligence, Engineering and Technology

Research Article

WRJAAIET-25-016

A Comparative Analysis of Machine Learning Models for URL-Based Phishing Detection

Rafi MRM*, Shaminda KAS, Nuski FAM, Amila Senarathne, Suhaif AM and Kanishka Yapa

Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

***Correspondence:** Rafi MRM, Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka, E-mail: it21300028@my.sliit.lk; DOI: https://doi.org/10.56147/aaiet.1.3.16

Citation: MRM R, KAS Shaminda, FAM Nuski, Senarathne A, AM Suhaif, et al. (2025) A Comparative Analysis of Machine Learning Models for URL-Based Phishing Detection. J Adv Arti Int Eng & Techn 1: 16.

Abstract

Phishing attacks pose a significant and ongoing cybersecurity threat, necessitating effective countermeasures. The challenge lies in accurately and automatically detecting malicious URLs, as traditional methods often fall short against evolving attacker techniques. This research addresses the need for improved detection by evaluating machine learning approaches applied to URL analysis. A dataset of labeled phishing and legitimate URLs, characterized by 30 distinct features encompassing lexical, host-based and content-related attributes, formed the basis of this study. Five machine learning models were trained and comparatively evaluated: Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), XGBoost (XGB) and a Stacking Classifier ensemble. Performance analysis revealed that the XGBoost classifier achieved the highest accuracy, correctly classifying approximately 97.4% of URLs in the test set. This study demonstrates the effectiveness of machine learning, particularly XGBoost, for high-accuracy phishing URL detection using comprehensive feature sets and contributes a functional prototype system demonstrating the approach.

Keywords: Phishing detection; URL analysis; Machine learning; Cybersecurity; XGBoost; Random forest; SVM; Stacking classifier; Feature engineering; Feature extraction; Classification; Malicious URL detection; Network security; Scikit-learn; Python

Received date: April 16, 2025; **Accepted date:** April 23, 2025; **Published date:** May 14, 2025

Introduction

The pervasive nature of the internet has revolutionized communication and commerce, yet it has also given rise to significant cybersecurity challenges. Among the most prevalent and damaging threats is phishing, a deceptive practice where attackers masquerade as trustworthy entities in electronic communications to illicitly obtain sensitive information such as login credentials, financial details and personal identifiers. These attacks often leverage social engineering tactics and sophisticated technical methods, resulting in substantial financial losses, identity theft and compromised organizational security. The scale and evolving complexity of phishing necessitate the development of robust, automated detection systems.

Uniform Resource Locators (URLs) are fundamental to web navigation and serve as the primary vector for directing users to malicious websites in many phishing campaigns. Attackers craft these URLs carefully, often embedding subtle indicators of fraud by mimicking legitimate domains, using obfuscation techniques like URL shortening or complex paths or exploiting structural anomalies. Consequently, the automated analysis of URL characteristics provides a critical early opportunity to identify and mitigate phishing threats before users interact with potentially harmful links.

Traditional anti-phishing methods, such as blacklist filtering and basic heuristic rule-based systems, face significant limitations. Blacklists are inherently reactive and often lag behind the rapid creation and disposal of phishing domains used in modern campaigns. Simple heuristics, while useful, can be easily bypassed by attackers aware of the rules and may struggle with the nuances of sophisticated attacks, leading to inadequate accuracy with high rates of false positives or false negatives.

Journal of Advanced Artificial Intelligence, Engineering and Technology

This paper addresses the need for more effective phishing detection by exploring the application of Machine Learning (ML) techniques to URL analysis. ML models offer the potential to learn complex patterns from diverse URL features, enabling more accurate and adaptive detection compared to static methods. The objective of this research is to develop and rigorously evaluate a high-accuracy phishing URL detection system by:

Utilizing a comprehensive set of 30 URL-based features encompassing lexical, host-based, content-derived and external reputation attributes;

Training and comparing five distinct ML classifiers (Decision tree, Random forest, Support Vector Machine, XGBoost and a Stacking ensemble) and;

Identifying the most effective model based on empirical performance evaluation.

This paper is structured as follows: Section 2 details the methodology, including dataset description, feature engineering, model implementation and evaluation metrics. Section 3 provides an overview of the system architecture and implementation. Section 4 presents and discusses the experimental results, comparing model performance and highlighting the best-performing classifier. Finally, Section 5 concludes the paper and outlines directions for future research.

Methodology

This section outlines the systematic approach employed for developing and evaluating the machine learning models for phishing URL detection. It covers the dataset used, the feature engineering process, data preprocessing steps, the machine learning algorithms implemented and the metrics used for performance evaluation.

Dataset

The study utilized a publicly available dataset sourced from Kaggle, commonly used for phishing website detection research (PhishingDataset.csv) [4]. This dataset comprises 11,055 URL instances, each labeled as either phishing (-1) or legitimate (1). Each instance is represented by 30 pre-extracted numerical features, designed to capture various characteristics indicative of a URL's legitimacy, plus the target label column ('Result'). An initial inspection confirmed the dataset was complete with no missing value.

Feature set

The detection models were trained using the 30 features provided in the dataset [4]. These features capture a diverse range of URL characteristics potentially indicative of phishing activity and are encoded numerically, primarily using values of -1 (suspicious), 0 (neutral/intermediate) and 1 (legitimate). The implementation logic for extracting these features from raw URLs is contained within a dedicated Python script (extract_script_py.py) developed as part of this project's broader scope. These features can be categorized as:

Lexical features: Attributes derived directly from the URL string, such as the presence of an IP address, URL length categorization, use of shortening services, presence of special characters ('@', '//', '-'), subdomain complexity (dot count) and inclusion of 'http'/'https' tokens within the domain/path.

Host-based features: Information related to the domain's registration and hosting environment, including domain registration duration, domain age, SSL certificate validity status, DNS record existence and abnormalities detected *via* WHOIS lookups.

C**ontent-based & external features:** Characteristics derived from analyzing basic webpage content or querying external services, such as the ratio of external links (anchors, images), usage of specific HTML tags (iframes, forms submitting to email, scripts), port status, redirect counts, website traffic rank, PageRank proxies (domain authority), Google indexing status and checks against statistical lists of known phishing domains/IPs.

This comprehensive feature set aims to provide the models with sufficient information to distinguish between phishing and legitimate URLs based on established indicators.

Experimental setup

The dataset was randomly partitioned into a training set (80%, 8844 samples) and a testing set (20%, 2211 samples) using Scikit-learn's train_test_split function [5] with a fixed random_state=12 to ensure reproducibility. Due to the nature of the features (encoded as -1,0,1), explicit feature scaling was deemed unnecessary and not applied. For compatibility with the XGBoost classifier, the target variable ('Result') was transformed from {-1,1} to {0,1}, where 0 represents Phishing and 1 represents Legitimate, for both training and testing sets.

Machine learning models

Five distinct machine learning classifiers were implemented and trained on the preprocessed training data:

Decision Tree (DT): Implemented using sklearn.tree.DecisionTreeClassifier with max_depth=5 to prevent overfitting.

Random Forest (RF): An ensemble model using sklearn.ensemble.RandomForestClassifier also with max_depth=5 for the base trees.

Support Vector Machine (SVM): Implemented using

sklearn.svm.SVC with a kernel='poly' and default regularization (C=1.0).

XGBoost (XGB): Implemented using the xgboost.XGBClassifier library. Hyperparameter tuning was performed (Section 3E), with the final model using parameters optimized for performance (*e.g.,* learning_rate=0.4, max_depth=7).

Stacking classifier: A multi-layer ensemble implemented with sklearn.ensemble.StackingClassifier, using RF, KNN and DT as first-layer estimators and another DT/RF stack followed by Logistic Regression as the final meta-learner.

All models were trained on the X_train, y_train data [5,6].

Evaluation metrics

Model performance was evaluated on the unseen test set using the following standard classification metrics from Scikit-learn [5]:

- Accuracy
- Precision
- Recall (Sensitivity)
- F1-Score
- Area Under the ROC Curve (ROC AUC)
- Confusion Matrix (analyzing TP, TN, FP, FN).

These metrics provide a comprehensive view of each model's effectiveness in correctly identifying both phishing and legitimate URLs.

System Architecture and Implementation Overview

This section describes the overall architecture designed for the phishing URL detection system and provides an overview of the implementation details, including the tools used and the integration of different components.

System architecture

The system architecture integrates offline model training and evaluation with an online prediction pipeline for classifying new URLs. The core components and data flow are depicted in **Figure 1**.

The architecture comprises:

Data preparation: Sourcing and preprocessing the dataset (as described in Section 2A, 2C) [4].

Model training & evaluation: Training multiple ML models (DT, RF, SVM, XGB, Stacking) in parallel on the training data and evaluating them on the test data using defined metrics (Section 2D, 2E).

Model selection & persistence: Selecting the best model (XGBoost based on evaluation) and saving the trained model object using pickle for deployment.

Feature extraction module: A dedicated module (extract_script_py.py) responsible for taking a raw URL input and generating the required 30-feature vector by executing various lexical, host-based, content-based and external checks (detailed in Section 2B).

Prediction engine: Loads the persisted XGBoost [6] model and uses it to classify the feature vector generated by the extraction module, outputting a 'Phishing' or 'Legitimate' prediction.

User interface (web/extension): A simple web UI developed to allow users to input a URL and initiate the detection process [8].

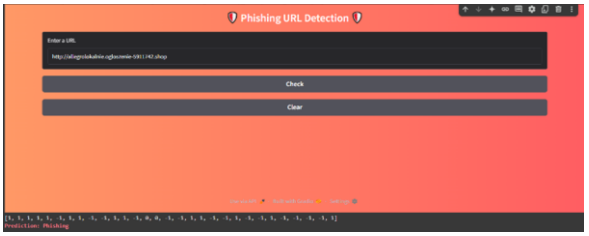

Figure 1: Interface to input URLs.

Results and Discussion

This section presents the empirical results obtained from evaluating the five machine learning models on the test dataset. The performance of each model is compared; the best performing model is analyzed in detail and the overall findings are discussed.

Model performance comparison

The performance of the Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), XGBoost (XGB) and Stacking Classifier models were evaluated on the held-out test set using the metrics defined in Section 2E. **Figure 2** summarizes the key performance results.

ML Model	Train Accuracy	Test Accuracy	Precision	Recall	f1 score	ROC_AUC
XGBoost	0.9888	0.9738	0.9714	0.9810	0.9762	0.9730
Stacking Classifier	0.9834	0.9634	0.9593	0.9744	0.9668	0.9622
Random Forest	0.9343	0.9331	0.9161	0.9661	0.9405	0.9296
Decision Tree	0.9268	0.9276	0.9108	0.9620	0.9357	0.9240

Figure 2: Model performance comparison.

As shown in **Figure 2**, the XGBoost classifier [6] significantly outperformed the other models across all

major metrics, achieving the highest Test accuracy, Precision, Recall, F1 Score and ROC AUC. The stacking classifier and SVM also demonstrated strong performance, surpassing the Decision Tree and Random Forest models implemented with limited depth **(Figure 3)**.

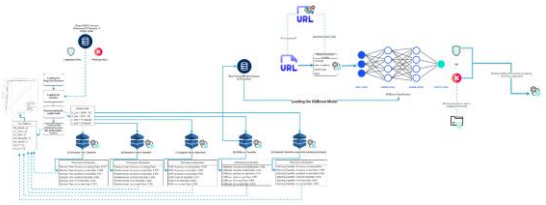

Figure 3: Overall system architecture.

ROC curve analysis

The Receiver Operating Characteristic (ROC) curves provide a visual comparison of the models' ability to distinguish between phishing and legitimate URLs across different thresholds. **Figure 4** displays the ROC curves for all five models plotted [9] against a baseline random classifier (TPR=FPR).

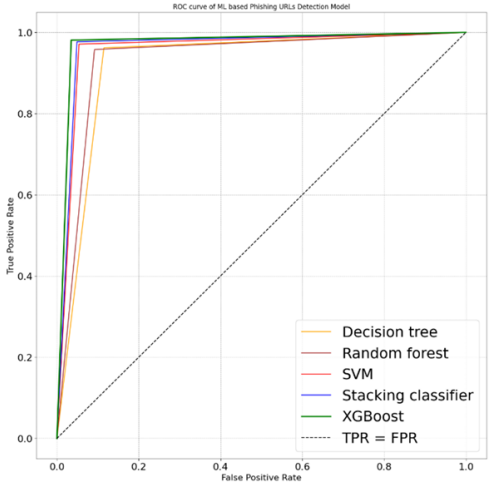

Figure 4: ROC curve plot.

The ROC curve analysis visually confirms the quantitative results presented in **Figure 4**. The XGBoost curve is positioned closest to the top-left corner, indicating superior performance with high True Positive Rates (Recall) and low False Positive Rates across various thresholds. The Stacking Classifier and SVM curves are also significantly above the baseline and the curves for the simpler decision tree and random forest models, demonstrating strong discriminative power. The Area Under the Curve (AUC) values numerically summarize this, with XGBoost achieving the highest AUC of 0.9730 [6].

Confusion matrix for the XGBoost model

True Negatives (TN): 966 (Correctly identified phishing URLs)

False Positives (FP): 35 (Legitimate URLs incorrectly flagged as phishing).

False Negatives (FN): 23 (Phishing URLs incorrectly identified as legitimate)

True Positives (TP): 1187 (Correctly identified legitimate URLs).

Feature importance

While detailed feature importance analysis varies slightly between models, examination of the importance scores generated by tree-based models like XGBoost consistently highlighted the significance of certain feature categories. Features related to SSL certificate status (SSLfinal_State), domain characteristics (age_of_domain, Domain_registeration_length), link structures within the page (URL_of_Anchor Links_in_tags) and external reputation metrics (web_traffic, Page_Rank) frequently appeared among the most influential predictors. This suggests that a combination of security indicators, domain trustworthiness signals, page structure analysis and external reputation is key to effective detection.

Conclusion and Future Work

This research successfully developed and evaluated a machine learning system for detecting phishing URLs, demonstrating the high efficacy of modern algorithms combined with comprehensive feature engineering for this critical cybersecurity task. Through a comparative analysis of five distinct classifiers trained on a dataset of over 11,000 URLs characterized by 30 diverse features, the XGBoost [6] model was identified as the most effective, achieving approximately 97.4% accuracy on the test set. This result, along with the strong performance of the Stacking Classifier and SVM, underscores the potential of machine learning to significantly enhance phishing detection capabilities beyond traditional methods. The project also involved the practical implementation of a feature extraction module and a prototype system, validating the feasibility of the approach.

The main contributions include a quantitative performance benchmark for contemporary ML models on this task and dataset, validation of XGBoost's high accuracy and the development of a proof-of-concept system. However, the study acknowledges limitations related to dataset representativeness and critically, the potential unreliability and latency associated with extracting certain features dependent on external APIs and web scraping in real-time, high-volume scenarios.

Future work should focus on several key areas to build upon these findings and address the limitations. Firstly, enhancing model robustness and performance could involve exploring different algorithms (*e.g.,* LightGBM,

deep learning sequence models like LSTMs), conducting more extensive hyperparameter tuning and incorporating techniques to handle potential class imbalance in real-world data. Secondly, significant effort is required in improving feature engineering, focusing on identifying a core subset of reliable, low-latency features, investigating dynamic analysis techniques, enhancing content analysis (NLP, visual similarity) and potentially replacing less stable free feature sources with commercial data feeds or more resilient scraping methods coupled with effective caching. Thirdly, expanding and continuously updating the dataset with more diverse and recent phishing examples is crucial for maintaining model relevance and improving generalization against zero-day attacks. Finally, advancing the system implementation towards a production-ready state necessitates deploying the model as a robust API, significantly improving the UI/UX with better explainability and feedback mechanisms, developing practical integrations like browser extensions and potentially implementing adaptive learning loops for ongoing model refinement based on new data and user feedback. Addressing the reliability of feature extraction remains a paramount challenge for transitioning such research into highly dependable, real-world security tools.

References

1. Filippi P, et al. (2019) An approach to forecast grain crop yield using multi-layered, multi-farm data sets and machine learning. Precis Agric 20: 1015–1029. [Crossref] [Google Scholar]
2. Dhanapal R, Ajanraj A, Balavinayagapragathish S, Balaji J (2021) Crop price prediction using supervised machine learning algorithms. J Phys Conf Ser 1916: 012042. [Crossref] [Google Scholar]
3. Izzo Z, Smart MA, Chaudhuri K, Zou JY (2021) Approximate data deletion from machine learning models. Proceedings of the 24th International Conference on Artificial Intelligence and Statistics 130: 1-9. [Google Scholar]
4. Kaggle (2024) Phishing website dataset.
5. Scikit-learn Developers (2024) Scikit-learn: Machine learning in Python.
6. (2024) XGBoost contributors, XGBoost documentation.
7. Keras Team (2025) Keras documentation.
8. Gradio Team (2025) Gradio: Build & share delightful machine learning apps.
9. Matplotlib Development Team (2024) Matplotlib: visualization with Python.
10. Waskom ML (2021) Seaborn: Statistical data visualization. J Open Source Softw 6: 3021. [Google Scholar]