

Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics

Ifeoluwa D. Oseni ^{1, 2, ✉}, Ilobekemen P. Oladoja ¹, Olumide Adewale ¹

1. Department of Computer Science, Federal University of Technology, Akure, Akure, NGA

2. Department of Computer Science, University of Ibadan, Ibadan, NGA

Received: June 03, 2026 | Review began: June 30, 2026 | Review ended: August 25, 2026 | Published: September 29, 2026

© Copyright 2026

This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

Phishing is a significant form of cybercrime. It exploits unsuspecting users through the use of deceptive websites designed to harvest sensitive information. This study proposes a practical hybrid phishing detection framework that integrates machine learning, explainable artificial intelligence, and a rule-based component specifically for handling inactive URLs using HTTP status heuristics. Random Forest (RF) and Extreme Gradient Boosting served as the baseline classifiers, with weighted soft voting and stacking ensembles applied to integrate their predictions and improve overall performance. The classifiers were trained on the 2024 Mendeley and UCI PhiUSIL phishing datasets. XGBoost achieved better accuracy, recall, and F1-score among the two baseline classifiers, while RF achieved better precision. Both ensemble methods surpassed the individual classifiers in accuracy, recall, and F1-score. The stacking ensemble, with a logistic regression meta-learner, slightly outperformed the weighted voting ensemble strategy, achieving an accuracy of 98.88%, recall of 99.01%, and an F1-score of 98.73%. To manage non-responsive links, an operational filter leverages HTTP status heuristics to assign a final classification. For URLs outside the 200-299 status range, this rule-based label takes precedence, while the underlying machine learning models concurrently generate a supplementary probability estimate using retrievable and imputed attributes. SHapley Additive exPlanations is used to interpret active URL model prediction and help highlight the most influential features that distinguish legitimate websites from phishing websites. This enhances transparency and also provides insights into phishing behaviors, such as suspicious domain structures and DNS anomalies. This proposed interpretable phishing detection approach is easy to use and deploy across diverse user groups, regardless of their level of technical expertise.

Categories: AI/ML-based decision support systems, Explainable AI, Machine Learning (ML)

Keywords: phishing, interpretable machine learning, xai, shap, model transparency, random forest, xgboost, soft voting, ensemble methods, http status code

Introduction

Phishing is a pervasive cyber threat that exploits human psychology through social engineering [1,2]. It makes use of deceptive emails and fraudulent websites to steal sensitive information [3]. Although phishing attacks can target any user, individuals with limited cybersecurity knowledge and busy decision-makers who cannot easily verify the legitimacy of a message or its sender are generally more vulnerable to such attacks. Users are vulnerable to phishing due to limited understanding of URLs and difficulty identifying legitimate websites. Redirections and concealed URLs, combined with time constraints, often lead to accidental access to fraudulent sites [2].

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

Traditional security measures often focus on technical vulnerabilities while a phishing attack uniquely targets the human element. This difference in priority has made it one of the most successful attack vectors since the existence of the Internet [4]. While machine learning (ML) has emerged as an effective method for detecting phishing URLs, these models typically operate as “black boxes,” providing predictions without explanations for users who require additional information to decide whether to trust a website link or not. This lack of interpretability undermines user confidence and may lead users to rely entirely on the model’s decision blindly, even though it is practically impossible for any model to achieve 100% accuracy in real-world applications. Furthermore, the ephemeral nature of phishing URLs presents an additional challenge, as a link’s HTTP status can change rapidly, rendering a purely ML-based classification obsolete. This research addresses these critical gaps by developing an interpretable and robust phishing detection system. This study uses two high-performing tree-based classifiers, which are Random Forest (RF) and Extreme Gradient Boosting (XGBoost), as the baseline predictive models. Furthermore, two ensemble strategies are developed using these classifiers to improve robustness, generalization capability, and overall detection performance. The SHapley Additive exPlanations (SHAP) framework, a model-agnostic explainable artificial intelligence (XAI) method, is used to quantify the global and local feature contributions influencing the predictive models for active URLs. To counter the short lifespan of phishing sites and get the real-time status of an individual URL, a rule-based component that verifies a URL’s active status via its HTTP response code was also implemented. The entire system is deployed as an interactive web application using Streamlit.

Related works

As phishing detection shifts toward ML-based solutions, several research priorities have emerged. These include optimizing baseline algorithm performance, refining feature selection methodologies, transitioning from theoretical models to live systems, and addressing the critical need for explainable phishing detection system.

A significant portion of research focuses on identifying the most effective ML algorithms. Almseidin et al. [5] reported that their RF model achieved an accuracy of 98.11% using 20 selected features out of the original 48 features on a balanced dataset of 10,000 websites. This superiority of tree-based models is supported by Butnaru et al. [6] who relied solely on lexical features (including novel metrics like Levenshtein distance). They found RF to be the best performer on their dataset of 100K URLs. Manoj et al. research showed that a Multilayer Perceptron (MLP) model slightly outperformed SVM and XGBoost using host-based, page-based, and lexical features, which indicates that optimal model selection can be context-dependent on the feature set [7]. Piñeiro and Portillo [8] found a simple Classification Tree to outperform both RF and SVM on their specific dataset. They introduced a valuable innovation by exploring logical conjunctions (“AND” and “OR”) of models to tailor performance towards high-phishing detection or low false-positive rates. The critical role of feature engineering is a consistent theme. Calzarossa et al. [9] provided a rigorous statistical analysis using Gini Index and Lorenz Zonoid to rank and select features. They identified the top six most important lexical features for phishing detection as URL length, path length, domain name length, filename length, number of vowels, and number of dots in the domain. Their work underscores that model performance is intrinsically linked to the quality and relevance of the input features.

Beyond pure accuracy, a key trend in recent literature is the development of deployable systems. Shombot et al. [10] developed a web application using Flask and MySQL. They integrated an SVM model and used a chi-square feature selection. Their work focuses on a continuous learning loop that integrates user interactions to enhance model performance and enable scalable, real-world applications. The most advanced step in this direction is the work of Misiek and Hyla [11]. They created a real-time Google Chrome extension specifically for detecting phishing within ‘https://t.co/’ redirection links on X (Twitter). They built the system by using a FastAPI backend and an XGBoost model that was 97.8% accurate. However, they only built the extension for one browser, and there was no model caching mechanism, which could lead to potential latency under load.

Maneriker et al. [4], developed URLTran, a transformer-based model for phishing URL detection. The researchers fine-tuned BERT and RoBERTa models and enhanced them with domain-specific vocabularies that were derived from a large collection of URLs. These vocabularies were constructed to capture lexical and structural patterns specific to URL data. The model was pre-trained using Masked Language Modeling and further strengthened through adversarial training

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

strategies, such as homoglyph substitutions and compound word manipulations, to improve robustness against evasion techniques. Experimental results showed that the URLTran BERT variant achieved a true positive rate of 86.80% at a false positive rate of 0.01% and outperformed convolutional neural network-based baselines such as URLNet under comparable evaluation settings.

An integrated model that combines MPNet, a transformer-based method for analyzing the semantic structure of URL sequences, with RasNet and TCMA modules for character- and word-level feature extraction was developed by Do et al. [12]. The model achieved a state-of-the-art detection rate of 99.71% on a dataset of one million samples and outperformed existing architectures by more than 1.36%.

Li et al. [13] proposed a privacy preserving and explainable phishing detection framework that integrates federated learning with multimodal large language models called FedPhishLLM. Qwen2-VL-2B model was fine-tuned and integrated with Low Rank Adaptation on decentralized client data. This enables collaborative model optimization without transferring raw user information to a central server. FedPhishLLM processes both visual features, such as webpage screenshots, and textual components like HTML content and URLs, directly on user devices to maintain data confidentiality. The proposed FedPhishLLM demonstrated strong generalization performance. The framework achieved an accuracy, precision, and F1 score of 95%, with a recall of 96%. It outperformed conventional convolutional neural network models in federated learning environments and demonstrated robustness against adversarial manipulations, including logo alteration attacks designed to bypass brand-based detection.

A comparative evaluation of lightweight transformer models and traditional ML classifiers for phishing email and URL detection was conducted by Tarapiah et al. [14]. Transformer models such as DistilBERT, ALBERT, and RoBERTa were compared with conventional ML classifiers such as RF, Logistic Regression, and Support Vector Machine. Tokenization was used for the transformer models, while TF-IDF vectorization was applied to the ML classifiers. The evaluation covered both balanced and imbalanced datasets and included phishing emails generated by language models. DistilBERT achieved the highest accuracy among the LLMs (98.44% on imbalanced email data), while RF demonstrated high computational efficiency with over 98% accuracy. The ML model performance increased while the LLMs slightly decreased when the dataset was balanced.

Uddin et al. [15] proposed an optimized and fine-tuned version of RoBERTa, a robust transformer-based architecture, for detecting phishing emails. The model achieved an accuracy exceeding 97% across both training and testing splits on balanced and imbalanced datasets. To enhance interpretability, the authors incorporated two XAI techniques, namely Local Interpretable Model Agnostic Explanations and Transformer Interpret. These XAI methods were used to compute positive and negative word-level attribution scores, which provided insight into the model's decision-making process.

As ML models have become more accurate, their "black box" nature has become a significant impediment to trust and adoption. Calzarossa et al. made strides in this area by integrating feature importance directly into their RF model, providing global explainability [9]. However, their work did not extend to local explainability that can justify individual URL predictions for end users. More recent phishing detection studies have incorporated SHAP, transformer-based models, and large language models to improve model interpretability [13,15,16]. While these approaches provide explanations for model predictions, they primarily focus on the explainability component. They do not integrate explainability with ensemble learning, lexical and network feature extraction, inactive URL handling, and deployment within a single phishing detection framework. This study addresses this by integrating RF and XGBoost with weighted and stacking ensembles, SHAP-based local and global explanations, HTTP status heuristics for inactive URLs, and an interactive phishing detection system. Owa and Adewole [17], after confirming RF's performance, explicitly noted the absence of real-time threat identification and a lack of investigation into hybrid models that could synergize the strengths of multiple algorithms. Similarly, the analysis of Misiek and Hyla [11] reveals gaps in building scalable, efficient backend systems capable of handling concurrent user requests and serving multiple browser ecosystems. While existing research provides a strong foundation of high-performing models, a significant gap remains for a holistic system that integrates four critical components:

1) a powerful ensemble model for superior accuracy,

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

2) robust, per-prediction explainability to overcome the “black box” problem,

3) a real-time mechanism to verify the active status of ephemeral URLs, and

4) a scalable architecture for cross-platform deployment.

Our work is explicitly designed to bridge this gap, delivering a transparent, practical, and resilient phishing detection solution.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

S. No.	Author	Dataset	Methodology	Result	Limitation
1	Almseidin et al. [5]	Phishing dataset that contains 10,000 samples (5,000 legitimate and 5,000 phishing)	Tested multiple machine learning models and feature selection techniques and selected J48, Random Forest, and Multilayer Perceptron which were evaluated using the top 20 selected features	Random Forest had the highest accuracy of 98.11% using the selected 20 features, it outperformed J48 and Multilayer Perceptron models	Limited validation on diverse real world datasets
2	Butnaru et al. [6]	A Kaggle dataset that contains 100,000 URLs (60,000 phishing and 40,000 benign) with features selected based on prior literature and novel additions like Levenshtein distance acquired from various sources such as PhishTank	Developed a lightweight phishing detection engine using supervised learning algorithms (Naive Bayes, Decision Tree, Random Forest, SVM, MLP) and compared performance against Google Safe Browsing	Random Forest performed best with 99.29% accuracy. It outperforming Google Safe Browsing which have an accuracy of about 45%	The evaluation was limited to datasets of phishing and benign URLs, which may not encompass all real-world scenarios or dynamic environments
3	Manoj et al. [7]	37,000 URLs (17,000 phishing and 20,000 benign) collected from personal web browser history, PhishTank, DMoz, and Alexa	Extracted host-based, page-based, and lexical features and applied classifiers like Logistic Regression, SVM, XGBoost, MLP, and AutoEncoders using WEKA	Multilayer Perceptron (MLP) achieved the highest accuracy of 85.8%, followed by SVM at 85.0% and XGBoost at 84.9%	Relies heavily on feature quality; detection may fail if attacks lack expected patterns (e.g., zero-hour attacks)
4	Piñeiro and Portillo [8]	Mendeley 2020 Phishing Dataset with 88,647 records (35% phishing, 65% non-phishing) and 111 extracted features	Developed a web architecture using Random Forest, Classification Trees, and SVM, along with logical conjunction	Classification Trees achieved the highest effectiveness with an average accuracy of 80% for both phishing and	Some dataset fields could not be calculated due to a lack of Python libraries; the dataset was limited to URLs and may not fully account

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

			models (AND/OR) for prediction	non-phishing predictions	for dynamic phishing scenarios
5	Calzarossa et al. [9]	10,000 URLs (5,000 phishing from PhishTank and 5,000 legitimate from Tranco) with 74 extracted features	Proposed an explainable machine learning model using Random Forest with a novel feature selection method based on Lorenz Zonoids to identify key features	The selected model with six key features achieved an AUC of approximately 0.96, comparable to the full model with 74 features but with reduced complexity	The methodology focuses on URL-based features and excludes visual or content-based features; the balanced dataset may not reflect real-world frequency
6	Shombot et al. [10]	Dataset of about 1,400 records (legitimate, suspicious, and phishing) with 9 features created by Abdelhamid et al. (2014)	Implemented a Support Vector Machine (SVM) model comparing Polynomial and Radial Basis Function (RBF) kernels, integrated into a web application with a database for self-improvement	The Polynomial kernel outperformed the RBF kernel, achieving an accuracy of 84.5% compared to 82.6% for RBF	The dataset size is relatively small (1,400 records), and the study lacks Explainable AI (XAI) techniques, making model decisions difficult for users to interpret
7	Owa and Adewole [17]	Combined dataset of over 640,000 URLs, including the Aalto University Dataset (96,018 URLs) and Kaggle dataset (549,346 URLs)	Conducted a comparative analysis of Random Forest, Support Vector Machine (SVM), and Decision Tree models using a rigorous dataset verification pipeline and advanced preprocessing	Random Forest achieved the highest accuracy of 87.85% on the Aalto dataset and 86.86% on the Kaggle dataset, outperforming SVM and Decision Trees	Lack of real-time threat identification and cross-domain validation; high-performing algorithms like Random Forest struggle with interpretability (black box issue)

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

8	Misiek and Hyla [11]	Dataset of URLs processed for feature extraction.	Created a browser plug-in using the XGBoost machine learning model to identify and flag malicious external URLs on web pages	The XGBoost algorithm demonstrated high performance in accurately identifying malicious URLs via the browser plugin	The plug-in was developed only for Google Chrome, lacks a caching mechanism for predictions, and cannot handle concurrent requests efficiently
9	Maneriker et al. [4]	Microsoft Edge & IE telemetry data (approximately 3.08 million URLs (1.04M train, 0.26M validation, 1.78M test))	Developed URLTran, a fine-tuned Transformer models (BERT, RoBERTa) on URL data using domain-specific vocabularies. Employed adversarial training with homoglyph substitutions and compound word splitting to improve robustness	Achieved a True Positive Rate (TPR) of 86.80% at a very low False Positive Rate (FPR) of 0.01%, outperforming CNN-based baselines (URLNet) by over 21%	The model showed significant performance drops against adversarial attacks prior to specific adversarial training, and pre-training from scratch on URLs required massive data to match fine-tuned NLP models
10	Do et al. [12]	Four datasets (Ebbu2017, PhishCrawl, 420K-PD, and 1M-PD).	Consists of 3 modules which are RasNet-TCMA-MPNet: A hybrid model combining RasNet (ResNet + embedding) for character features, TCMA (TCN + Multi-Head Attention) for feature selection, and MPNet (Transformer) for semantic analysis	Achieved a detection rate of 99.71% on the 1M-PD dataset	The architecture involves three distinct deep learning modules running in parallel, which implies higher computational complexity and resource consumption compared to simpler models.
11	Li et al. [13]	Putra et al., 2023 Phishing website dataset which contains 5,244 legitimate and 5,151 phishing sites	Developed FedPhishLLM, a federated Learning framework using Qwen2-VL-2B (Vision-Language Model). Used Low-	FedPhishLLM outperformed all baseline models under FL and had an accuracy, precision and F1-Score of 95% with a	The pre-trained VLM showed poor generalization (53% accuracy) without fine-tuning. While client-side

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

			Rank Adaptation (LoRA) to fine-tune the model on decentralized client data (HTML, URLs, screenshots)	recall value of 96% Fine-tuned model significantly outperformed the pre-trained model which had an accuracy of 53%	inference is lightweight (approximately 28MB), the backend infrastructure requires high-performance GPUs (32GB+ video memory) for training and aggregation
12	Tarapiah et al. [14]	Manually curated multi source email and URL dataset comprising 26,365 cleaned email records, of which 29% were labeled phishing, and 67,391 cleaned URL records, of which 23% were labeled phishing	Benchmarked lightweight LLMs (DistilBERT, ALBERT) using sub-word tokenization against traditional ML classifiers (Random Forest, SVM) using TF-IDF vectorization. The study evaluated performance across both balanced and imbalanced datasets of emails and URLs	Random Forest had the highest accuracy among the machine learning classifiers more than 98% accuracy, while DistilBERT was the best large language model (more than 98%). Balanced datasets improved ML model performance but slightly reduced LLM accuracy	LLMs required significantly higher training time (hours vs. minutes) and their accuracy dropped slightly when the dataset was balanced which indicates sensitivity to synthetic data
13	Uddin et al. [15]	A Kaggle Phishing Email dataset which has 11,322 Safe Emails and 7,328 Phishing Email	Fine-tuned RoBERTa for text classification. Integrated LIME and Transformer Interpret to compute positive/negative word attribution scores for transparency	Achieved 98.45% testing accuracy on the balanced dataset and 97.96% on the imbalanced dataset	While LIME and Transformers Interpret offer transparency, their different underlying mechanisms can lead to opposite polarities (positive vs. negative) being assigned to the same word. This inconsistency can confuse end-users. The study addresses this limitation by proposing LITA (LIME-Transformer Attribution), a

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

					hybrid approach that balances and reconciles these conflicting scores
--	--	--	--	--	---

TABLE 1: Summary of Literature Review

URL, Uniform Resource Locator; SVM, Support Vector Machine; MLP, Multilayer Perceptron; WEKA, Waikato Environment for Knowledge Analysis; XGBoost, Extreme Gradient Boosting; AUC, Area Under the Curve; RBF, Radial Basis Function; XAI, Explainable Artificial Intelligence; IE, Internet Explorer; CNN, Convolutional Neural Network; TCN, Temporal Convolutional Network; MPNet, Masked and Permuted Pre-training for Language Understanding; PD, Phishing Dataset; LLM, Large Language Model; VLM, Vision-Language Model; FL, Federated Learning; HTML, Hypertext Markup Language; GPU, Graphics Processing Unit; TF-IDF, Term Frequency–Inverse Document Frequency; LIME, Local Interpretable Model-Agnostic Explanations; LITA, LIME-Transformer Attribution

Table 1 is the summary of the related works.

Materials And Methods

The study was implemented using RF and XGBoost as the baseline classifiers. They served as the foundation for a weighted voting ensemble and a stacked ensemble. These ensemble approaches were designed to leverage the complementary strengths of the base classifiers to improve the accuracy and robustness of phishing detection. Figure 1 depicts the flow chart.

How to cite this article:

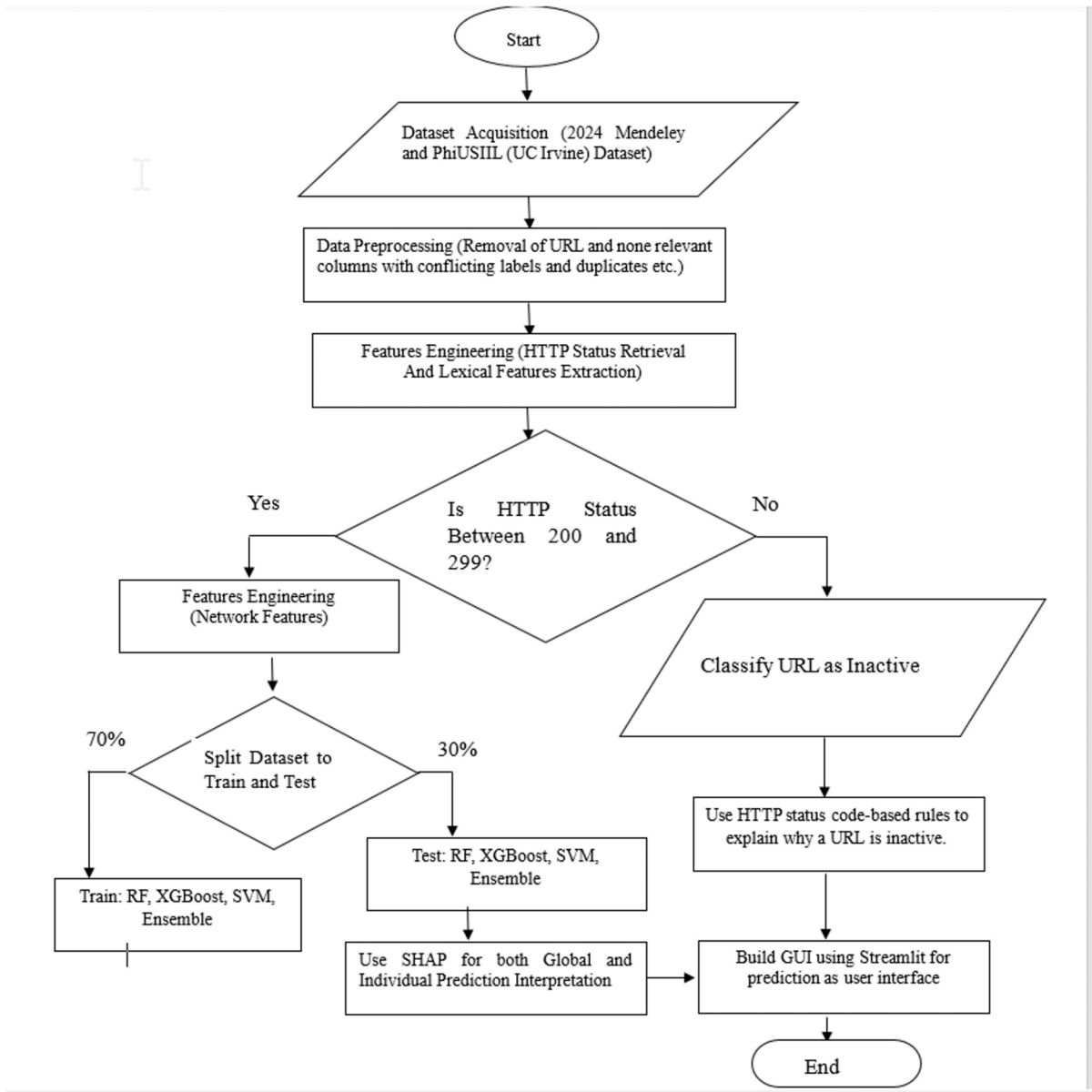


FIGURE 1: Flow Chart

SVM, Support Vector Machine; XGBoost, Extreme Gradient Boosting; SHAP, SHapley Additive exPlanations; RF, Random Forest; GUI, Graphical User Interface

This research work was carried out in five phases which are as follows:

1. Data Acquisition.
2. Data preprocessing.
3. Feature engineering.
4. Model development.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

5. Model evaluation.

Data acquisition

Two contemporary, large-scale 2024 datasets were used for this research work. They are PhiUSIIL (UC Irvine) and Mendeley Phishing Dataset. These sources were selected based on their large-scale and unbalanced nature, which provides an imbalanced benchmark setting that is closer to operational class imbalance than a deliberately balanced evaluation dataset. The combined number of URLs for the datasets are 685,971 URLs with the Mendeley dataset having over 450K URLs.

Data preprocessing

A multi-stage preprocessing pipeline was designed for this phase. The goal of this phase is to have a refined, conflict-free dataset optimal for feature engineering.

The first phase involved schema standardization to ensure consistency across both sources. The type column was renamed to label, and the values legitimate and phishing were mapped to 0 and 1, respectively, for the Mendeley dataset. Only the raw URL and label columns were retained for use for the PhiUSIIL dataset while the 54 derived columns from the URL were discarded in favor of other lexical and network related features that were extracted during this research work. This was done to ensure the same features were extracted from both datasets.

A global conflict-resolution process was then applied to scan both datasets for duplicate URLs to identify label inconsistencies. URLs with conflicting labels across either dataset or both were removed while a single instance of duplicate URLs with identical labels was retained in the first dataset while others were removed. This phase led to the removal of 3,815 URLs.

Feature engineering

A total of 127 features were extracted from each URL in three stages. The first two stages require internet access. To manage execution time and network overhead, the extraction was distributed across local and cloud environments, specifically GitHub Actions and Azure DevOps. The process involved HTTP requests, DNS record resolution, SSL certificate inspection, and page-level content parsing.

GitHub Actions workflows were defined using YAML and executed on Ubuntu virtual machines under the free tier allocation of one parallel job up to 1,800 minutes per month, in addition to a self-hosted runner configured on a local workstation for extended runs. Azure DevOps pipelines were also used, running on the Azure Pipelines 2 agent with the ubuntu 24.04 runner image version 20250609.1.0 and agent version 4.255.0. This distributed execution strategy was selected because the feature extraction process requires repeated HTTP requests, DNS resolution, SSL certificate inspection, and webpage parsing, all of which are network-dependent operations. Executing these tasks across GitHub Actions, Azure DevOps, and a self-hosted runner enabled concurrent processing of independent URLs, reduced the overall extraction time, and accommodated the computational requirements of the complete dataset. This configuration improved resource utilization, minimized delays associated with sequential network-bound processing, and enabled extraction of all 127 features for every URL while remaining within the available platform limits. The combination of lexical, host-based, certificate, DNS, and webpage content features provides complementary information that improves the representation of phishing and legitimate websites for subsequent classification.

The feature extraction implementation was done with Python versions 3.10 and 3.11. Pandas was used for data handling, httpx and asyncio for asynchronous HTTP processing, SSL for certificate inspection, and playwright for browser-based fallback resolution. Concurrency was limited to 10 parallel requests with a 15 second timeout, and processing was performed in batches of 500 URLs to ensure execution stability.

No external DNS or SSL APIs were used, and no third-party API keys were required, as all lookups relied on native Python libraries. However, a repository secret was configured in GitHub to securely enable the workflow to upload generated artifacts. To further maintain stability, the dataset was partitioned into segments of approximately 10,000 URLs per run and later merged after validation, with detailed logging maintained throughout the extraction process.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

HTTP Status Retrieval Stage

A custom asynchronous script extracted live HTTP status features including HTTP status response code, redirect behavior, and activity status for each URL. To handle the scale and complexity of the task, the implementation leveraged concurrent network requests (using asyncio and httpx) and browser emulation (via playwright) python packages.

Network-Related Features Extraction

A total of 54 network and page-based features were extracted to capture technical and structural characteristics of each URL. These features included domain record analytics (counts of NS, MX, A, SPF, DKIM, and DMARC records), SSL certificate details, and domain age. Additionally, page content features were extracted, including the presence of key HTML elements such as titles, images, buttons, password inputs, submit inputs, JavaScript, favicons, and anchor tags. These elements collectively help characterize both the infrastructure and composition of each web page, providing discriminative signals for classification. This was done for URLs whose HTTP status codes were between 200 and 299.

Lexical Features Extraction

A total of 68 lexical features were extracted from each URL to capture syntactic patterns commonly associated with phishing. These included structural properties such as URL length, domain length, and subdomain depth, as well as counts of specific characters including hyphens, question marks, dollar signs, equal signs, slashes, and underscores.

$$\mathbf{S} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p]^T \quad (1)$$

The feature extraction function is given in Equation (1), where $\mathbf{S}$ denotes the feature vector containing the features extracted from a URL, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p$ represent the extracted features from a URL, and T denotes the transpose of the feature vector.

Model development

The classification framework operates using a dual-layer approach that combines a heuristic rule-based filter with binary ML inference. The system performs binary phishing classification for active websites, combined with a rule-based operational-status outcome to categorize URLs as legitimate (0), phishing (1), or inactive (2). This categorization is determined primarily by the HTTP operational state of the link.

$$\mathbf{P} = \{2, \text{ if } \text{HTTP}(\mathbf{s}) < 200 \text{ or } \text{HTTP}(\mathbf{s}) > 299 \hat{y}_{ml}(\mathbf{x}) \in \{0, 1\}, \text{ if } 200 \leq \text{HTTP}(\mathbf{s}) \leq 299\} \quad (2)$$

If a URL returns a status code outside the 200 to 299 range, it is automatically flagged as inactive. The inactive classification serves as an operational filter rather than a ML output, processing non-responsive links before machine learning inference. For all active URLs within the specified status range, the final classification is determined by the output of the ML ensemble. This logic is formally defined in Equation (2) where $\mathbf{P}$ denotes the final predicted class assigned to a URL $\mathbf{s}$, $\mathbf{x}$ represents the extracted features, and $\hat{y}_{ml}(\mathbf{x})$ is the predicted output.

The prediction $\hat{y}_{ml}(\mathbf{x})$ in equation (2) outputs a binary value in $\{0,1\}$ for active URLs. Dataset $\mathcal{D}$ consists of N number of URLs set and is a pair $(\mathbf{x}_i, \mathbf{y}_i)$. The ML phase utilizes two primary base learners, RF, and XGBoost, which are evaluated both individually and through integrated ensemble strategies for binary phishing classification. Each classifier is trained on the Dataset $\mathcal{D}$ extracted features.

For the RF model, the final probability estimate for the phishing class is derived by averaging the probabilistic predictions of all individual decision trees within the ensemble. This is formally defined in Equation (3), where T is the total number of trees and $p_t(\mathbf{y} = 1 | \mathbf{x})$ represents the probability predicted by the t -th decision tree for input $\mathbf{x}$.

$$P_{RF}(\mathbf{y} = 1 | \mathbf{x}) = \frac{1}{T} \sum_{t=1}^T p_t(\mathbf{y} = 1 | \mathbf{x}) \quad (3)$$

How to cite this article:

More broadly, let $P_{ml}(y = 1 | x)$ denote the continuous probability output derived from any of the ML base classifiers. The final binary prediction, $\hat{y}_{ml}(x)$, is obtained by applying a predefined decision threshold of 0.7 to this probability, as formally defined in Equation (4).

$$\hat{y}_{ml}(x) = \{1, \text{ if } P_{ml}(y = 1 | x) \geq 0.7, 0, \text{ otherwise}\} \quad (4)$$

XGBoost ($f_{XGB}(x)$) in Equation (5) builds an additive model of K regression trees, where $\mathcal{F}$ is the space of regression trees and $f_k(x)$ is the prediction of the k th regression tree for input x .

$$f_{XGB}(x) = \sum_{k=1}^K f_k(x), \quad f_k \in \mathcal{F} \quad (5)$$

The model is trained by minimizing the binary logistic (cross-entropy) loss (L) in Equation (6), where N is the total number of training samples, y_i is the true class label, and p_i is the predicted probability for the i th sample.

$$L = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (6)$$

The probability for the phishing class (p_i) derived from Equation (7) is obtained by applying the sigmoid function $\sigma(\cdot)$ to XGBoost output ($f_{XGB}(x_i)$).

$$p_i = \sigma(f_{XGB}(x_i)) = \frac{1}{1 + \exp(-f_{XGB}(x_i))} \quad (7)$$

For a new input x , Equation (8) computes the probability of the phishing class ($y = 1$), and the final binary prediction is subject to Equation (4).

$$P_{XGB}(y = 1 | x) = \frac{1}{1 + \exp(-f_{XGB}(x))} \quad (8)$$

Two ensemble strategies were developed and compared for the phishing detection system. They are a weighted soft-voting ensemble, which consists of both RF and XGBoost classifiers, and a stacked ensemble that uses a logistic regression meta-learner to optimally combine the probabilistic outputs of the base classifiers. The aim of the ensemble methods is to develop a strong classifier that can leverage the strength of the two strong baselines (RF and XGBoost).

The weighted ensemble method combines the probabilistic outputs of the base classifiers using pre-defined weights of both baseline classifiers. Its formula is expressed in Equation (9) below, where W_{RF} and W_{XGB} represent the weights for RF and XGBoost classifiers; P_{RF} , P_{XGB} are the predicted probability of phishing for a sample x from RF and XGBoost, respectively. Ensemble weights of 0.35 for RF and 0.65 for XGBoost, alongside a 0.70 classification threshold, were selected using validation data exclusively within the training partition. The final test set was reserved strictly for unbiased performance estimation and remained untouched until all hyperparameters and ensemble decisions were finalized.

$$P_{\text{weighted ensemble}}(x) = W_{RF} \cdot P_{RF} + W_{XGB} \cdot P_{XGB} \quad (9)$$

The same threshold and final binary prediction used for this ensembled model are in Equation (4).

For the stacking ensemble approach, a logistic regression meta-learner was trained on the out-of-fold (OOF) predictions of the base models generated through a 5-fold stratified cross-validation procedure. This procedure reduces overfitting and allows the meta-learner to learn the optimal combination of base model predictions. The out-of-fold probabilities of RF and XGBoost serve as the input features for the meta-learner, which is then trained on the true labels to produce the final stacked predictions.

How to cite this article:

Let $P_{RF}(y = 1 | x)$ and $P_{XGB}(y = 1 | x)$, denote the phishing probabilities of the phishing class from the RF and XGBoost base models, respectively, while w_{RF} and w_{XGB} denote the learned weights assigned to each base model output, with b representing the bias term. The sigmoid function $\sigma(z)$ maps the linear combination to a probability value. The stacked ensemble probability is defined in Equation (10), where the meta-learner optimally integrates the base model predictions. A decision threshold of 0.7 is applied to these probabilities to obtain the final binary classifications (see Equation (4))

$$P_{\text{stack ensemble}}(y = 1 | x) = \sigma(w_{RF} \cdot P_{RF}(y = 1 | x) + w_{XGB} \cdot P_{XGB}(y = 1 | x) + b) \quad (10)$$

The contribution of the feature extraction pipeline was evaluated through an ablation study that compared lexical, network, and combined feature sets. The statistical significance of the proposed stacking ensemble relative to the baseline classifiers was evaluated using McNemar's test. To quantify the statistical reliability and uncertainty of the performance estimates, 95% confidence intervals (CIs) were determined for all primary evaluation metrics. Non-parametric bootstrapping was performed using 1,000 resampled iterations of the test dataset ($N=80,824$) with replacement, calculating the 2.5th and 97.5th percentiles to establish the lower and upper bounds without imposing parametric distributional assumptions. All binary classifications were evaluated using the predefined decision threshold of 0.70.

Model evaluation and interpretability

To enhance model transparency, SHAP is integrated into the study. SHAP is a game-theoretic interpretation method that assigns each feature a contribution value toward a specific prediction (in our case phishing). Let the phishing classifier be denoted as f in Equation (11). The prediction for instance x is expressed as:

$$f(x) = \phi_0 + \sum_{i=1}^M \phi_i \quad (11)$$

Where ϕ_0 is the baseline prediction, defined by the expected model output over the dataset, M is the total number of features, and ϕ_i represents the contribution of feature i to the prediction for instance x . For local explanation, the Shapley value of feature i for model f and instance x is given in Equation (12) where F is the full set of features, S is a subset of features not containing i , and $f_S(x)$ denotes the model prediction using only the features in subset S . This approach allows each prediction to be decomposed into a baseline and feature contributions, providing a clear quantitative explanation of each feature's effect on the prediction.

$$\phi_i(f, x) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x) - f_S(x)] \quad (12)$$

SHAP values rely on the model's internal processing of extracted features, which presents a challenge for inactive URLs. When a URL returns an HTTP status code outside the 200 to 299 range, it is classified as inactive (predicted_label = 2) via the operational filter, and this rule-based label determines the final output. The base and ensemble models still compute a supplementary phishing probability from retrievable attributes for reference, but SHAP-based explanations are bypassed for these predictions, and the system relies on the HTTP status code itself to justify the inactive classification.

To maintain clarity, the system provides a rule-based explanation centered on the HTTP status code and its implications instead of using SHAP waterfall or force plots. This ensures that the user receives a direct, logical justification based on the operational state of the URL when a full feature-based explanation is not feasible.

How to cite this article:

Results And Discussion

Experimental setup

The system was developed using a combination of a personal computer (HP Latitude 7410 with an Intel Core i7-10610U processor, 16 GB RAM, and 512 GB solid-state storage, running Windows 11 Pro 64-bit) and two free-tier cloud-based platforms (GitHub Actions and Azure DevOps pipelines) to support the extraction of network-related features due to the size of the entire datasets and the need for faster extraction of the network features.

After the datasets had been acquired, the next phase, which is the data preprocessing phase involves standardizing the URL and label column names across both datasets to ensure consistency. The value 1 is assigned to phishing URLs and 0 to benign or legitimate URLs for the dataset where the label column values are not already encoded. Duplicate URLs were identified within each dataset individually, and a cross-dataset comparison validation was done. URLs that appeared with conflicting labels, either within the same dataset or across datasets, were completely removed. For duplicates with consistent labels, only the first occurrence was retained, while subsequent entries were discarded. To maintain transparency and reproducibility, all duplicate URLs were documented in a separate CSV file, including the number of occurrences and the dataset in which each duplicate was found. 3,815 URLs with conflicting labels were identified to be duplicates and handled through this process.

The next phase is the feature extraction phase, network-related features are extracted in two stages the first being the HTTP status stage, where the HTTP status and redirect status (has_redirect) status for each URL are extracted. A by-product of this stage is the is_active column, which has a default value of 0 for URLs that are not within the http status range of 200 to 299. A custom script was developed and run on the personal PC but due to the estimated time of completion being long, a GitHub action workflow was created to help extract this information which required the provisioning of an Ubuntu virtual machine environments. The workflow was configured with scheduled and manual triggers and executed within the GitHub Actions maximum runtime limit of six hours per job. Python version 3.11 was used for this feature extraction phase, with support from libraries such as pandas, requests, python-whois, dnspython, tldextract, beautifulsoup4, urllib3, aiohttp, aioudns, and argparse.

The second stage of the feature extraction process extracts all other network related attributes for URLs that were confirmed to be active (within the range of 200 to 299). Due to the size of the dataset and the computational overhead associated with DNS resolution, WHOIS queries, SSL inspection, and content-level analysis, this phase was executed using a hybrid infrastructure that combined Azure DevOps pipelines with the local computing environment. The primary workload was distributed across Azure DevOps pipelines to leverage parallel processing capabilities and reduce overall execution time. Feature extraction was also performed locally on the HP Latitude 7410 PC, particularly in instances where pipeline execution was interrupted, reached runtime limits, or required artifact continuation. In such cases, intermediate outputs were retrieved and processing was resumed on the local machine to ensure completion and prevent data loss. Azure DevOps pipelines were configured to run on the Azure Pipelines 2 agent, utilizing the ubuntu 24.04 runner image, version 20250609.1.0, with agent version 4.255.0. The dataset produced from the HTTP status stage was partitioned into multiple subsets to enable parallel execution across independent jobs. Each job operated within a six-hour runtime limit in accordance with platform constraints. Python 3.10 was provisioned and dependencies were installed from a dedicated requirements file to maintain consistency, and the same custom network feature extraction script was used across both cloud and local platforms.

The extracted features were written to separate CSV files for each batch, resulting in feature-enriched outputs that included DNS-based attributes (A, MX, TXT, NS), email authentication indicators (SPF, DKIM, DMARC), WHOIS metrics (domain age and registration period remaining), SSL certificate properties (validity, expiry, self-signed checks, hostname matching), and HTTP/HTML characteristics (redirects, response metadata, security headers, cookies, and structural elements). After all batches were processed, the batches that were earlier split into multiple batches were then merged back to give a consolidated CSV with the extracted network-related features by concatenating the batches in sequential order based on their filename indices, which would subsequently pass through a custom script that would also append the lexical features to the CSV. The final output then becomes the input for the ML model.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

The setup for the ML model begins with integrating the final Mendeley and PhiUSIIL datasets with the extracted features. Only rows with HTTP status in the range of 200 to 299 are passed to the ML model. Columns with more than 50% missing values are dropped, while columns that were not dropped whose values were not extracted are imputed with the column mean value from the filtered subset. Non-informative columns such as URLs, source identifiers, and error messages are excluded from the feature set, with the label column being the target variable.

Training and testing splits are performed separately for each class to maintain exact 70:30 proportions. RF and XGBoost models are initialized with fixed random states for reproducibility, and an ensemble is configured through weighted averaging by combining the models' probabilities. A logistic regression meta-learner is also prepared for stacking, trained on out-of-fold predictions to prevent information leakage.

SHAP is incorporated for model interpretability. A subset of the training data serves as the background for XGBoost's TreeExplainer, allowing computation of feature contributions in log-odds and probability terms. The base log-odds and corresponding probability provide a reference for model predictions, while mean absolute SHAP values rank features by their influence on the phishing class. Visualization and summary tables are prepared to highlight the top predictive features, with log-odds and probability impacts displayed.

A real-time feature extraction module was developed to compute the same lexical and network attributes used during model training and evaluation for an individual user-submitted URL. The module adheres strictly to the original feature schema, ensuring structural consistency between offline experimentation and live prediction. Each submitted URL is first parsed and normalized in a standardized manner before feature computation to maintain alignment with the preprocessing procedures applied during dataset construction.

Lexical features capture structural properties of the URL string, while network features are obtained dynamically through DNS resolution, WHOIS queries, SSL certificate inspection, and HTTP and HTML analysis. All features are mapped directly to the predefined training schema before prediction. A heuristic verification layer checks whether the submitted URL already exists in the reference dataset and informs the user of its recorded phishing status, while still generating a model-based probability estimate. For URLs with inactive HTTP status codes, the system flags their operational state as inactive but proceeds with prediction using all retrievable attributes, ensuring that a valid phishing probability is produced in every case.

The final dataset used for this research work consists of 129 columns with 86 distinct HTTP Status Codes extracted from the two datasets, with values ranging between 0 and 999. The majority of the extracted codes conformed to the standard definitions of the Hypertext Transfer Protocol while several non-standard or custom status codes were also identified like code 999, which might indicate the implementation of a proprietary response mechanism by certain websites, extending beyond the officially recognized specifications.

Python was used to implement the proposed models. Model performance was evaluated using four standard classification metrics: Accuracy, Precision, Recall, and F1 Score, as defined in Equations (13) to (16).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (13)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (14)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (15)$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (16)$$

The evaluation metrics were computed from the confusion matrix, where true positives (TP) and true negatives (TN) denote correctly classified phishing and legitimate Uniform Resource Locators, respectively, while false positives (FP) and false negatives (FN) denote misclassified legitimate and phishing Uniform Resource Locators.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

Results

Figure 2 shows the process of extracting lexical and network-related features from a URL.

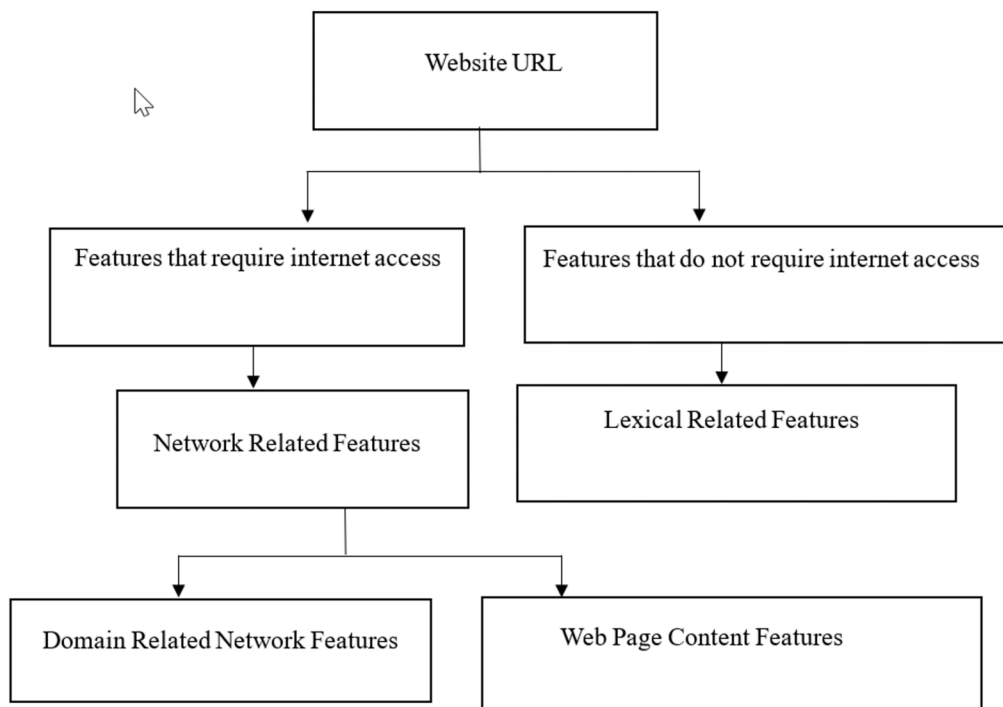


FIGURE 2: Feature Extraction Process for Website URL Phishing Detection for Lexical and Network-Related Features

Of the initial 685,971 URLs, 20,176 were removed during data cleaning due to duplicate entries (16,361) and conflicting labels (3,815), resulting in a final dataset of 665,795 unique URLs. In all, 269,411 (40.5%) of the unique 665,795 URLs had an HTTP status code in the range of 200 to 299. They were used for the ML model development and divided into training (70%) and testing (30%) sets. To ensure data quality and minimize noise, 41 columns that were missing more than half of their values were dropped. Most of these columns were page-related attributes like title, iframe, image tags, and SSL details. Their absence is likely caused by the increasing use of content delivery networks such as Cloudflare, which often verify traffic sources before serving full page content. This filtering ensured that the model was trained on reliable and consistently available features, reducing noise from sparsely populated variables. Also, four non-predictive columns (url, source, http_status, and label) were excluded prior to model training, resulting in a final set of 84 features passed and used for model training. This final set consists of all 68 original lexical features and 16 network features, as the remaining 38 network variables were excluded due to an excessive proportion of null values.

How to cite this article:

Model	Accuracy (95% CI)	Precision (95% CI)	Recall (95% CI)	F1-Score (95% CI)
Random Forest	0.9872 [0.9865, 0.9879]	0.9859 [0.9846, 0.9872]	0.9849 [0.9837, 0.9862]	0.9854 [0.9845, 0.9863]
XGBoost (Baseline)	0.9877 [0.9869, 0.9884]	0.9849 [0.9835, 0.9862]	0.9870 [0.9858, 0.9882]	0.9859 [0.9851, 0.9869]
Weighted Ensemble (Random Forest 35% + XGBoost 65%)	0.9879 [0.9872, 0.9887]	0.9855 [0.9842, 0.9868]	0.9871 [0.9859, 0.9883]	0.9863 [0.9854, 0.9872]
Stacking Ensemble (LR meta)	0.9888 [0.9882, 0.9896]	0.9845 [0.9832, 0.9859]	0.9901 [0.9891, 0.9911]	0.9873 [0.9866, 0.9882]

TABLE 2: Performance Comparison of Base Classifiers and Ensemble Methods on the Test Set (N = 80824, Threshold = 0.70)

The evaluation metrics and their corresponding 95% bootstrap confidence intervals for all evaluated models on the active URL test set ($N = 80,824$) are presented in Table 2. Across all configurations, the narrow confidence intervals indicate relatively low uncertainty in the estimated test-set metrics under the bootstrap procedure. Between the two baseline models, XGBoost achieved an accuracy of 98.77% (95% CI: [0.9869, 0.9884]) and a recall of 98.70% (95% CI: [0.9858, 0.9882]), marginally outperforming RF, which achieved an accuracy of 98.72% (95% CI: [0.9865, 0.9879]) and a recall of 98.49% (95% CI: [0.9837, 0.9862]). RF, however, maintained a higher precision of 98.59% (95% CI: [0.9846, 0.9872]). The weighted soft-voting ensemble yielded modest performance gains (F1-score of 98.63%). The Stacking Ensemble (with a logistic regression meta-learner) achieved the highest overall detection performance, attaining an accuracy of 98.88% (95% CI: [0.9882, 0.9896]), a recall of 99.01% (95% CI: [0.9891, 0.9911]), and an F1-score of 98.73% (95% CI: [0.9866, 0.9882]). These tight confidence bounds reinforce the McNemar's test results in Table 4, confirming that the performance gains of the stacking ensemble over the baseline classifiers are statistically significant.

The evaluation metrics for all models are given in Table 2. The results show that all models achieved strong predictive performance, with accuracies above 98.72%. Among the two base classifiers, XGBoost slightly outperformed RF in accuracy, recall, and F1-score, while RF had marginally higher precision. The combination of the two classifiers in a weighted ensemble provided modest improvements in accuracy, F1-score, and recall, reflecting the benefit of averaging predictions. The Stacking Ensemble, however, clearly outperformed all other models. It had the highest accuracy (98.88%), recall (99.01%), and F1-score (98.73%), indicating that integrating base learners through a logistic regression meta-learner not only balances the strengths of individual classifiers but also reduces the impact of weaker predictions, producing a more reliable and robust detection system. Its precision, however, was slightly lower (98.45%) than that of the other models, which indicates a modest increase in false positives despite the overall gains in recall and F1-score.

To quantify the contribution of the proposed feature extraction pipeline, an ablation study was conducted using lexical features only, network features only, and the combined feature set. The results are presented in Table 3.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	PR-AUC
Lexical (RF)	0.9851	0.9833	0.9826	0.9830	0.9953	0.9895
Lexical (XGB)	0.9870	0.9817	0.9887	0.9852	0.9963	0.9932
Lexical (Weighted)	0.9872	0.9826	0.9883	0.9854	0.9963	0.9932
Lexical (Stacking)	0.9868	0.9819	0.9882	0.9850	0.9964	0.9933
Network (RF)	0.7645	0.8375	0.5747	0.6817	0.8753	0.8218
Network (XGB)	0.7484	0.8307	0.5356	0.6513	0.8649	0.8093
Network (Weighted)	0.7574	0.8448	0.5478	0.6646	0.8745	0.8229
Network (Stacking)	0.7535	0.8344	0.5466	0.6605	0.8688	0.8154
Combined (RF)	0.9872	0.9859	0.9849	0.9854	0.9974	0.9948
Combined (XGB)	0.9877	0.9849	0.9870	0.9859	0.9981	0.9971
Combined (Weighted)	0.9879	0.9855	0.9871	0.9863	0.9982	0.9972
Combined (Stacking)	0.9888	0.9845	0.9901	0.9873	0.9981	0.9970

TABLE 3: Feature Ablation Study Results (Threshold = 0.70)

ROC-AUC, Receiver Operating Characteristic–Area Under the Curve; PR-AUC, Precision–Recall Area Under the Curve; RF, Random Forest; XGBoost, Extreme Gradient Boosting

In Table 3, lexical features alone achieved high predictive performance across all classifiers, with the weighted ensemble obtaining the highest accuracy of 98.72%, while the stacking ensemble achieved the highest ROC-AUC of 99.64% and PR-AUC of 99.33%. In contrast, models trained using only network features produced substantially lower performance, with the RF achieving the highest accuracy of 76.45% and F1-score of 68.17%. The combination of lexical and network features produced the best overall performance, with the stacking ensemble achieving an accuracy of 98.88%, recall of 99.01%, and F1-score of 98.73%. These results indicate that lexical features contain most of the discriminative information, while network features provide complementary information that improves phishing detection when both feature groups are combined.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

Comparison	Discordant (Base Correct, Stack Wrong)	Discordant (Base Wrong, Stack Correct)	Statistic	p-value	Significant ($\alpha = 0.05$)
Random Forest vs Stack	59	191	68.6440	1.18×10^{-16}	Yes
XGBoost vs Stack	102	198	30.0833	4.14×10^{-8}	Yes
Weighted Ensemble vs Stack	59	131	26.5316	2.59×10^{-7}	Yes

TABLE 4: McNemar's Test Comparing the Proposed Stacking Ensemble With the Baseline Classifiers (Decision Threshold = 0.70)

To assess whether the observed performance differences were statistically significant, McNemar's test was performed to compare the proposed stacking ensemble with the baseline classifiers using the same test set and a decision threshold of 0.70. The test evaluates discordant predictions between paired models by comparing instances correctly classified solely by the baseline classifier (Base Correct, Stack Wrong) against those correctly classified solely by the stacking ensemble (Base Wrong, Stack Correct). As shown in Table 4, the proposed architecture achieved more correct discordant predictions than RF (191 versus 59), XGBoost (198 versus 102), and the weighted ensemble (131 versus 59). The corresponding McNemar statistics yielded p-values well below the 0.05 significance level for all comparisons, indicating that the observed improvements over the baseline models are statistically significant. The Confusion Matrix for the RF is shown in Figure 3.

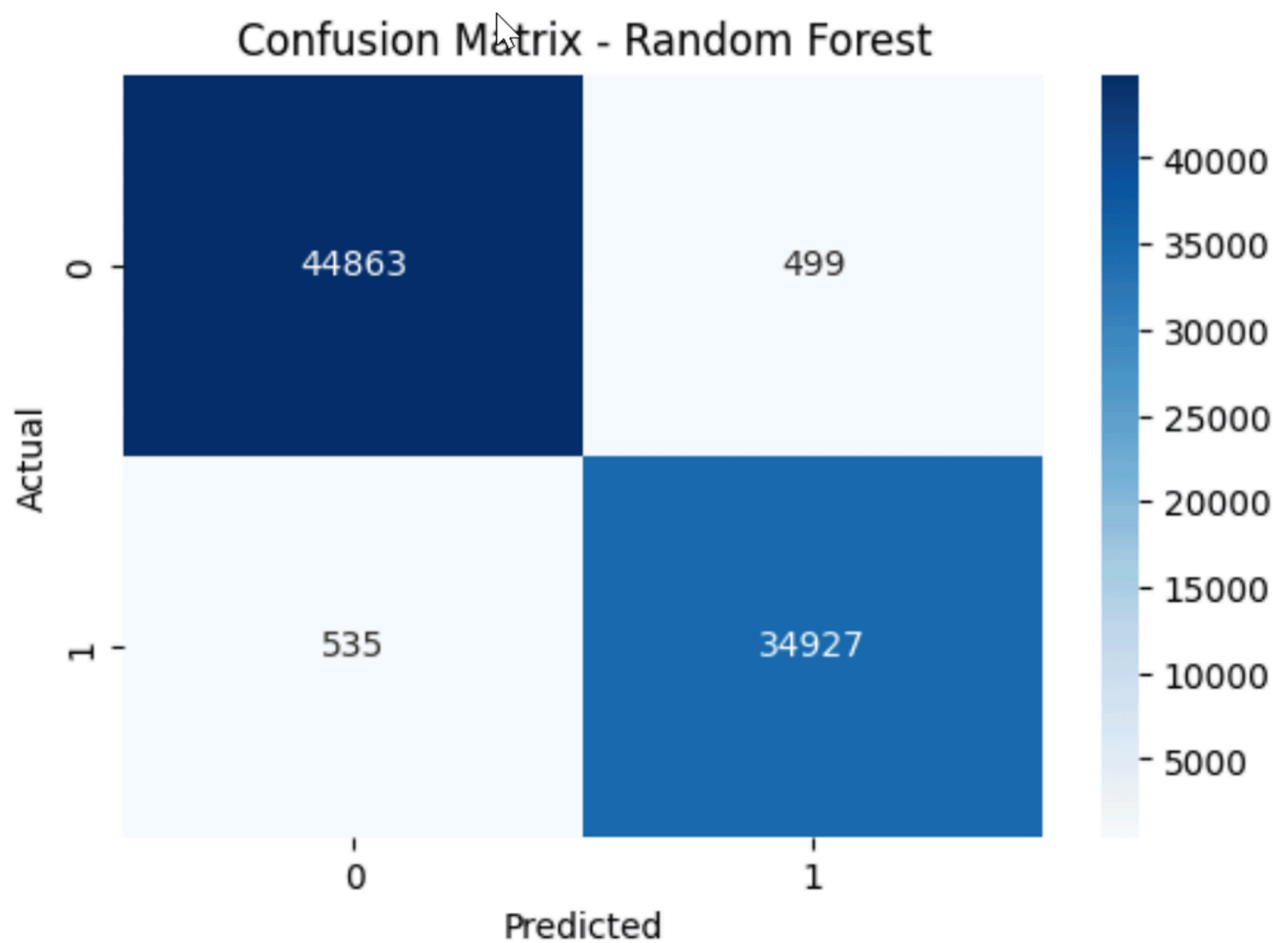


FIGURE 3: Random Forest Model Confusion Matrix

Confusion Matrix for XGBoost model is shown in Figure 4.

How to cite this article:

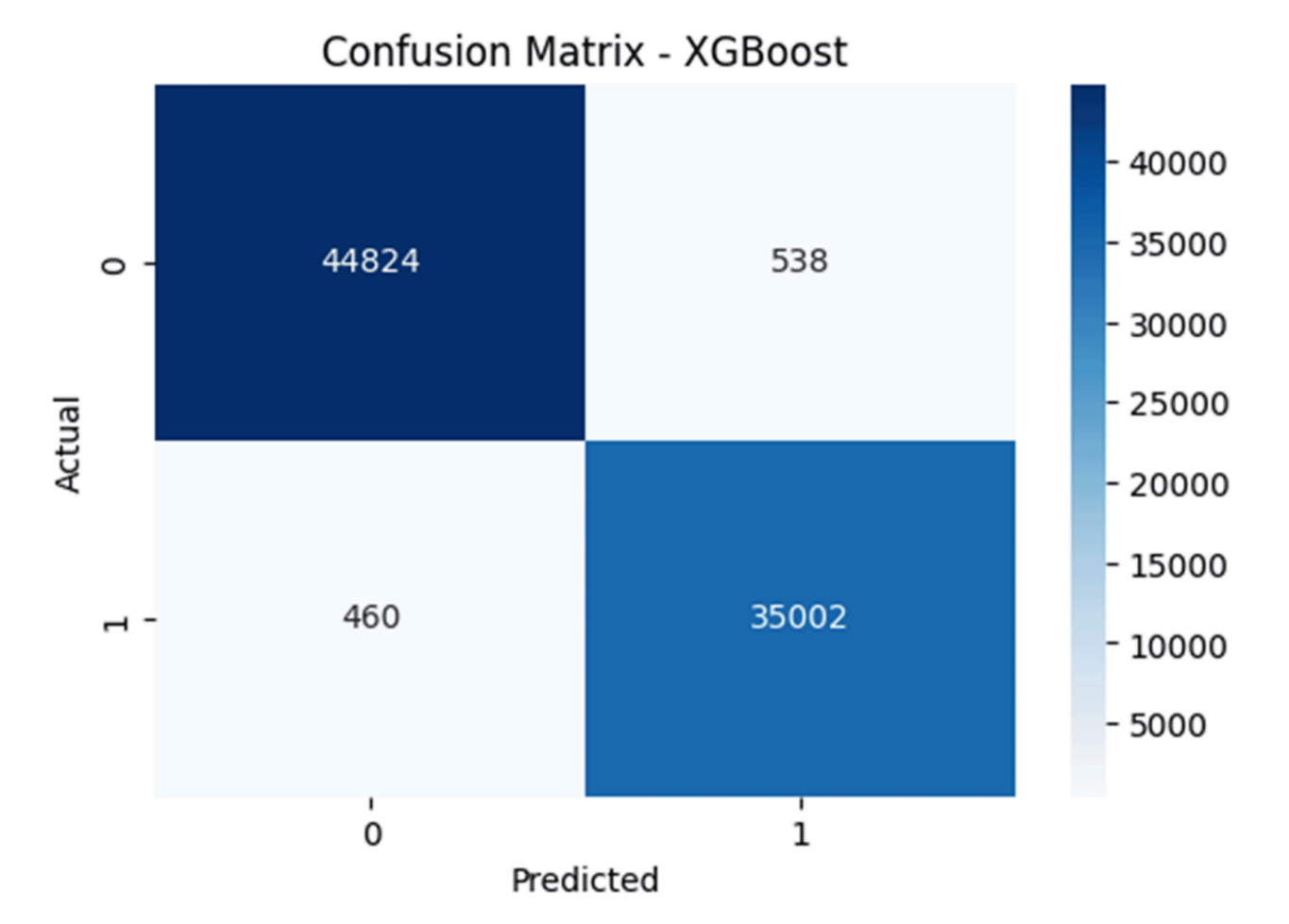


FIGURE 4: Confusion Matrix of XGBoost Model

XGBoost, Extreme Gradient Boosting

Confusion Matrix of the weighted ensemble is shown in Figure 5.

How to cite this article:

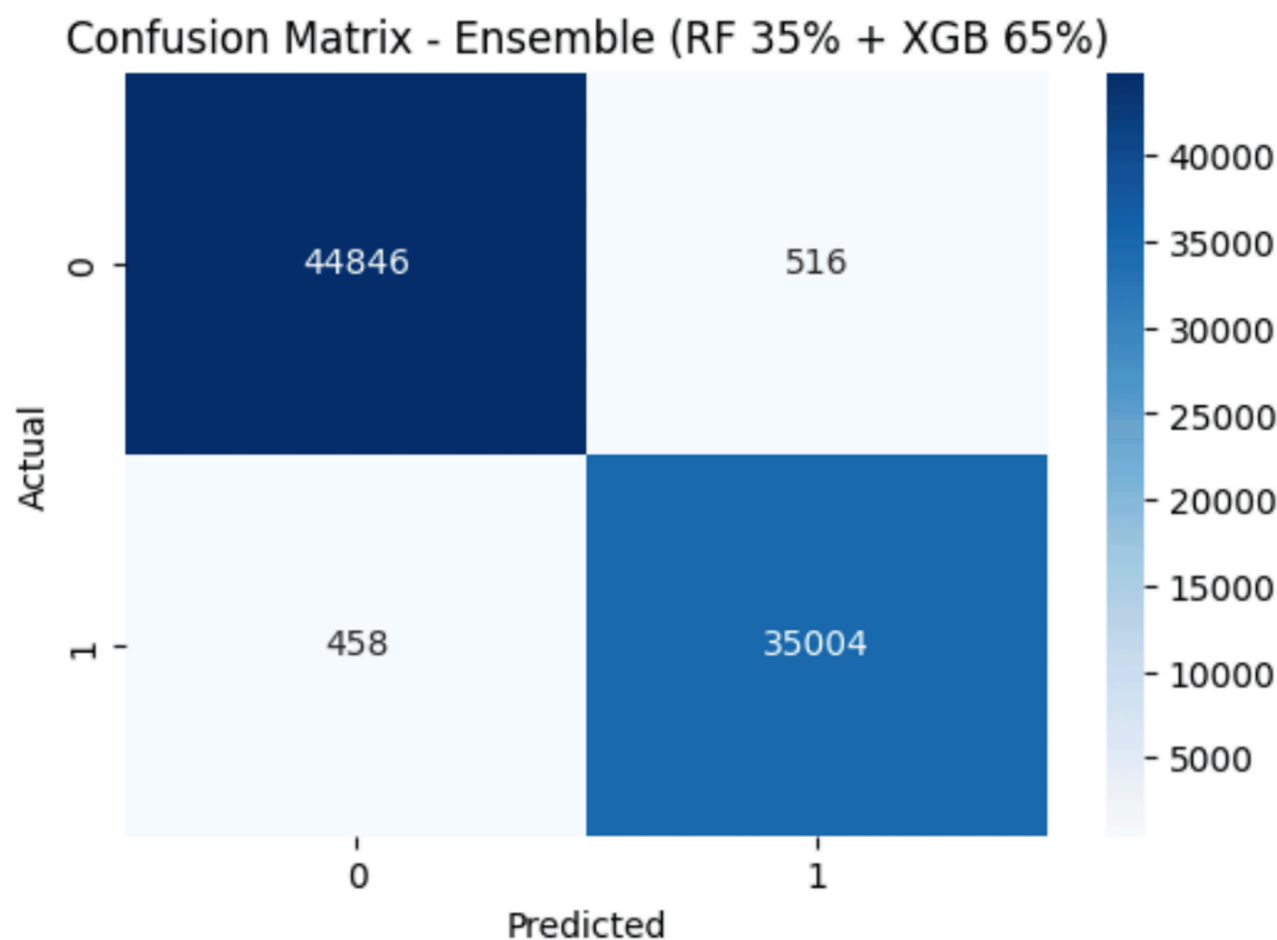


FIGURE 5: Confusion Matrix of the Weighted Ensemble (Random Forest 35% + XGBoost 65%)

RF, Random Forest; XGBoost, Extreme Gradient Boosting

Confusion Matrix of the stack ensemble is shown in Figure 6.

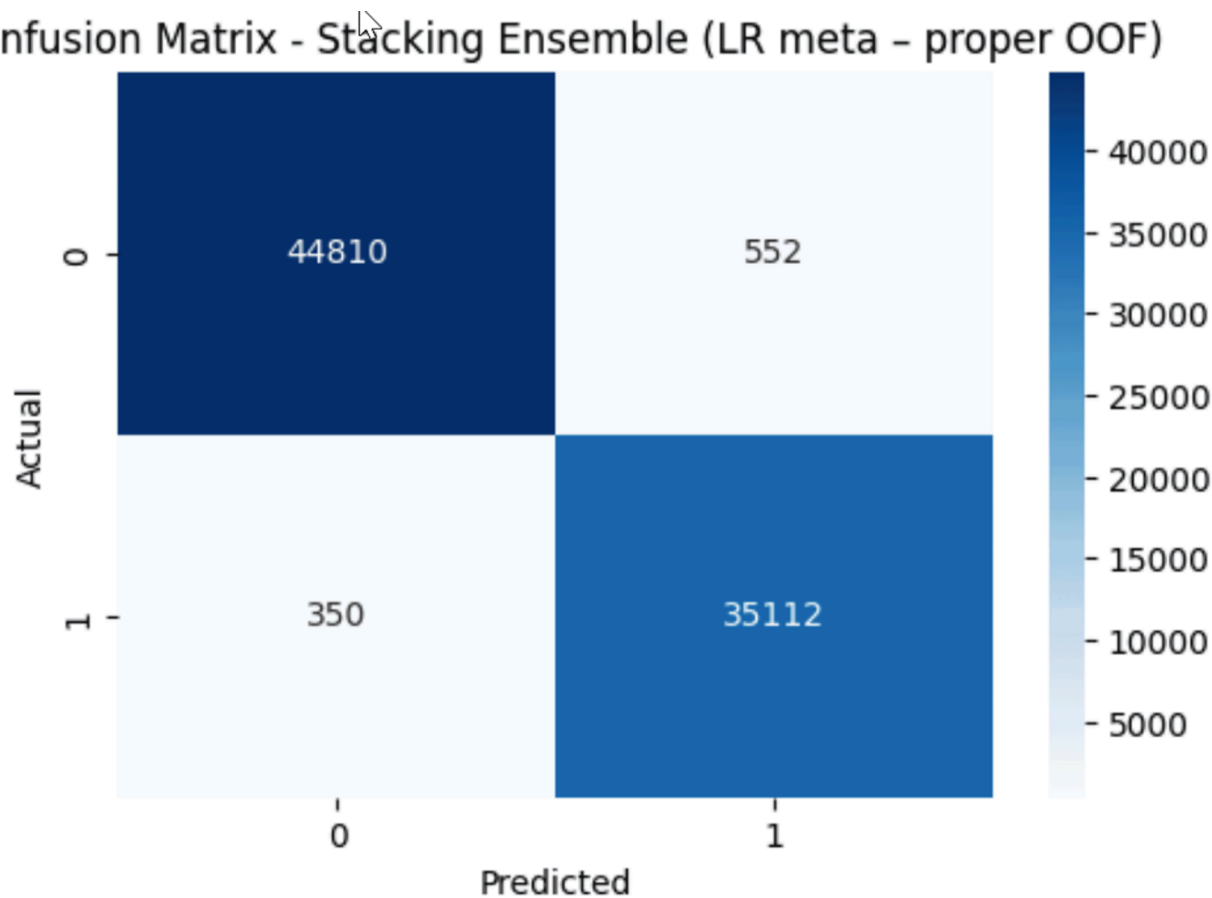


FIGURE 6: Confusion Matrix of the Ensembled Model

After the model has been developed, the model with its components is saved with Joblib and loaded to the Streamlit Application designed for end users to get predictions for an individual URL, which makes use of the ML model, extract all features from the URL, and uses a rule-based explanation for URLs whose http status code is not between 200 and 299 while using the ML model for URL within the status code range as well as SHAP for the individual URL. Each model (RF, XGBoost, Weighted Ensemble, and Stack Ensembled) is loaded for users. Users are able to see the prediction score of the selected model as well as the top 10 features that increased or decreased its probability. If a user inputs a URL that already exists in the dataset, they are notified of its presence along with its phishing status in the response they get, while a prediction is simultaneously generated for active URLs. Figures 7 to 10 show the prediction results for an active and inactive URL both within the dataset and outside the dataset with the stacked ensemble ML model.

How to cite this article:

Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics

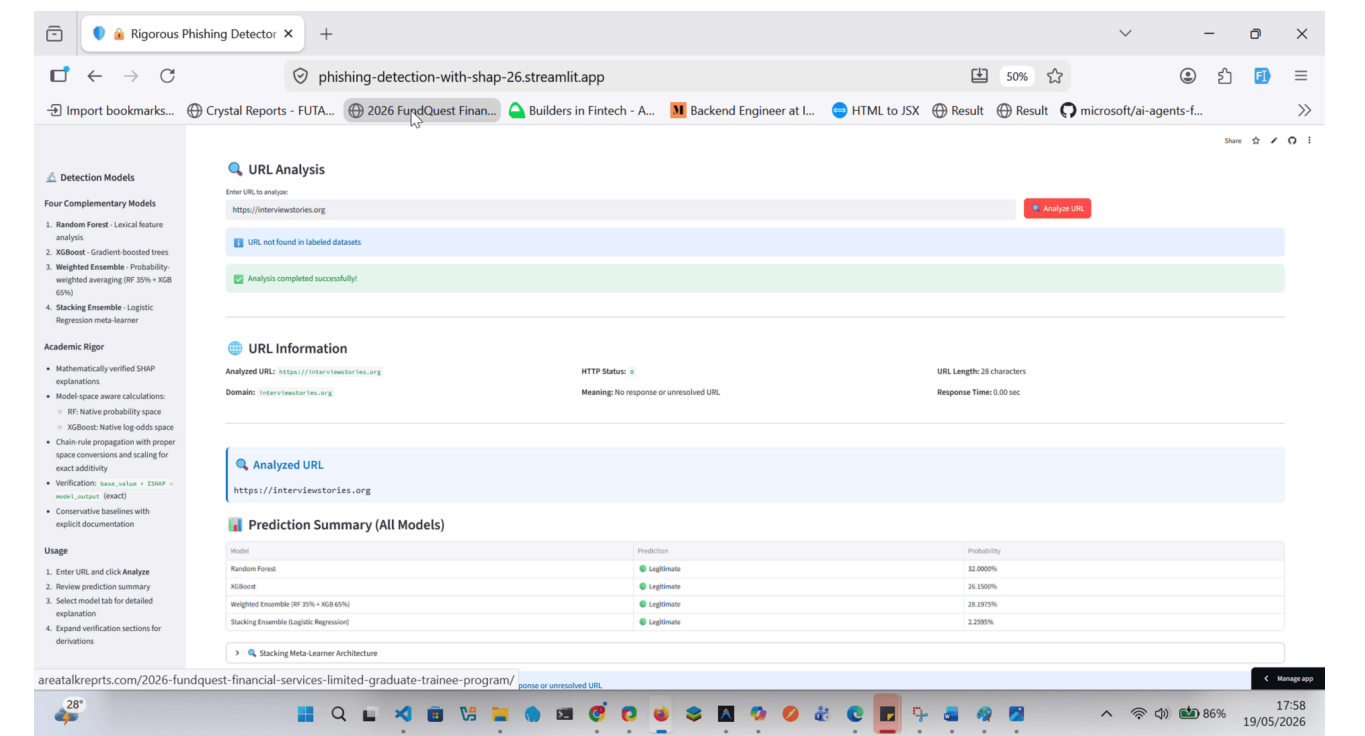


FIGURE 7: Phishing Prediction Response with HTTP Status Code Advisory for Inactive URL (https://interviewstories.org)

Figure 8 shows part of the data returned for an active URL (https://www.skincareliving.com) while Figure 9 shows the waterfall and force plots of the Stacking Ensemble (Logistic Regression Meta-Learner) model.

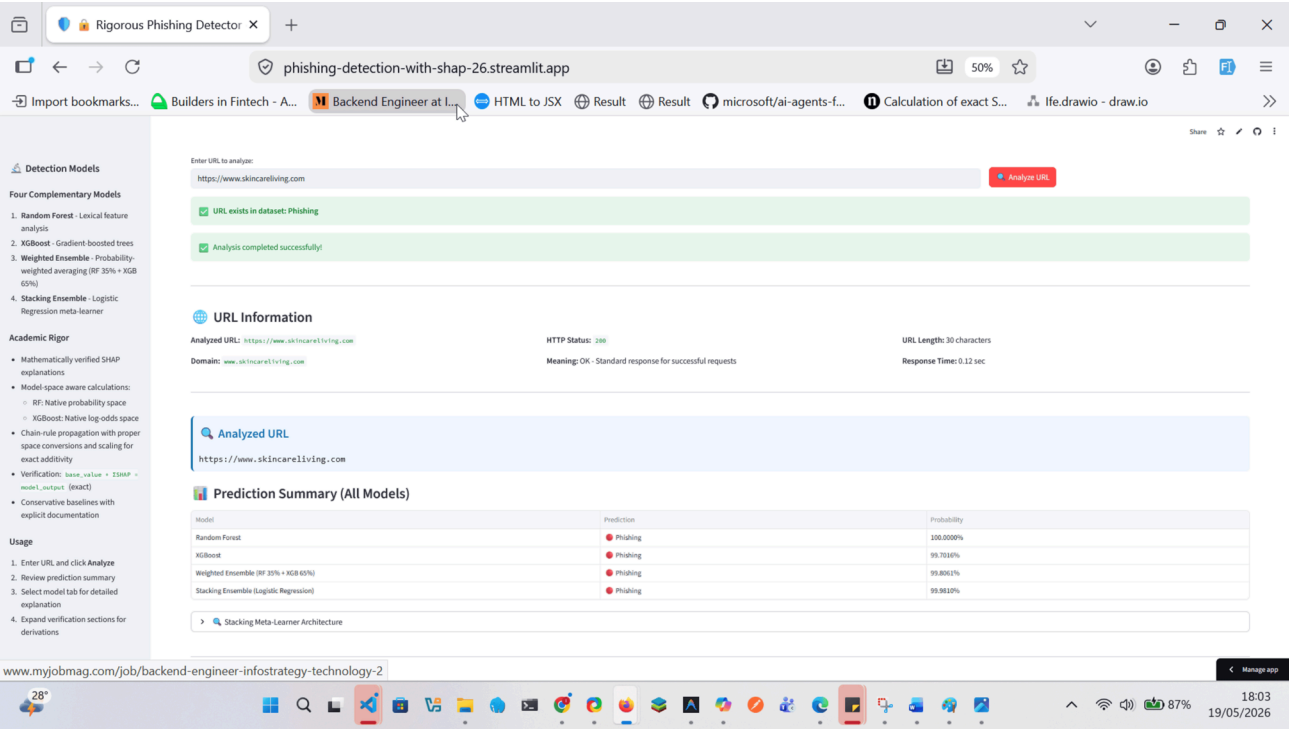


FIGURE 8: Phishing Prediction Response with HTTP Status Code Advisory for Active URL (<https://www.skincareliving.com>)

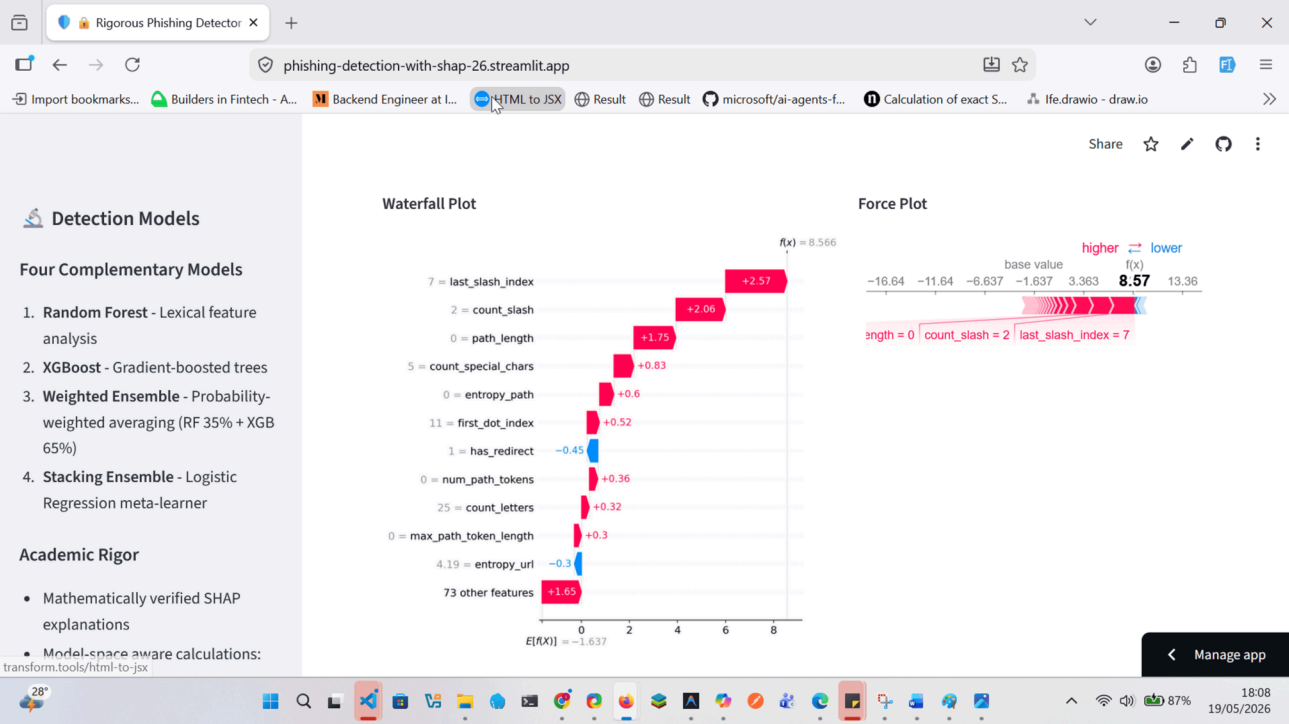


FIGURE 9: Waterfall and Force Plot Explanation for the Stacking Ensemble Model for Active URL (<https://www.skincareliving.com>)

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

For the general dataset, the Beeswarm SHAP explanation is used. Figure 10 shows the Beeswarm global explanation plot for the Stack Ensemble (LR meta) model.

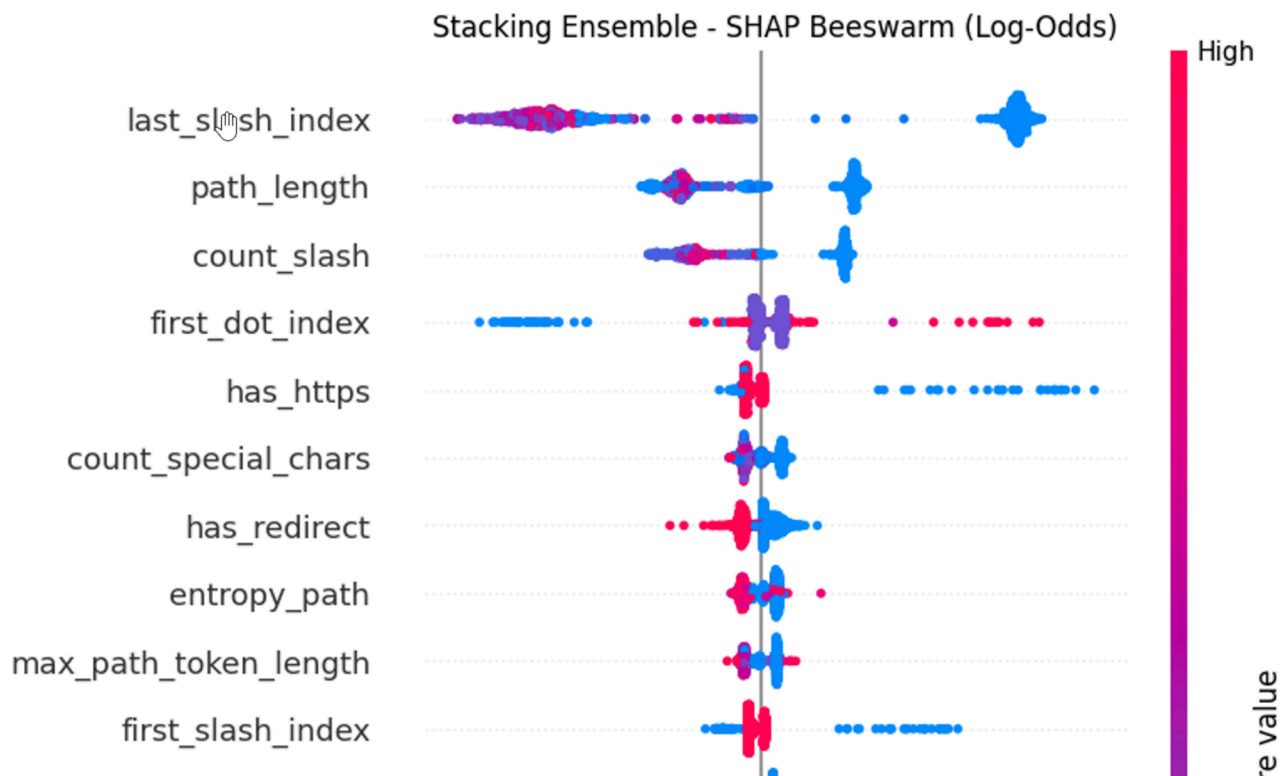


FIGURE 10: Top 10 Features Ranked by Importance Using Stacking Ensemble Method

Model	Output Space	Expected Output	Predicted Probability	Dataset Phishing Rate
Random Forest	Probability	0.438618	0.438618	0.438752
XGBoost	Log-odds	-0.246227	0.438752	0.438752
Weighted Ensemble (Random Forest 35%, XGBoost 65%)	Probability	0.438705	0.438705	0.438752
Stacking Ensemble (Logistic Regression Meta-Learner)	Log-odds	-0.246584	0.438664	0.438752

TABLE 5: Model Predictions and Baseline Comparison for SHAP-based Interpretability

Table 5 compares the expected output values, $E[f(x)]$, and final predicted probabilities of the evaluated models against the baseline dataset phishing rate of 0.438752. While the predicted probability estimates closely approximate the empirical prevalence of phishing within the dataset (ranging from 0.438618 to 0.438752), the expected output values are presented in the native mathematical space of each respective model and should therefore not be misconstrued as universally comparable probabilities. This indicates that the models and ensemble approaches maintain predicted probabilities consistent with the observed class distribution.

Rank	Stacking Ensemble (Log-Odds, Prob)	Weighted Ensemble (Probability)	XGBoost (Log-Odds, Prob)	Random Forest (Probability)
1	last_slash_index (2.0169, 0.4577)	last_slash_index (0.2767)	last_slash_index (4.5967, 0.3678)	last_slash_index (0.1078)
2	path_length (0.7275, 0.1746)	path_length (0.0416)	has_https (0.6977, 0.0165)	count_slash (0.0846)
3	count_slash (0.6481, 0.1548)	count_slash (0.0291)	first_dot_index (0.6371, 0.0229)	path_length (0.0808)
4	first_dot_index (0.2341, 0.0535)	first_dot_index (0.0225)	has_redirect (0.3602, 0.0235)	count_special_chars (0.0253)
5	has_https (0.1439, 0.0325)	has_redirect (0.0198)	path_length (0.2323, 0.0210)	first_dot_index (0.0224)
6	count_special_chars (0.1407, 0.0317)	has_https (0.0145)	dns_txt_count (0.2026, 0.0119)	entropy_path (0.0218)
7	has_redirect (0.1365, 0.0308)	count_special_chars (0.0089)	dns_ns_count (0.1869, 0.0097)	first_slash_index (0.0175)
8	entropy_path (0.1327, 0.0299)	dns_txt_count (0.0089)	dns_a_count (0.1532, 0.0061)	max_path_token_lengt h (0.0174)
9	max_path_token_lengt h (0.1284, 0.0289)	entropy_path (0.0074)	count_letters (0.1438, 0.0075)	has_https (0.0157)
10	first_slash_index (0.1169, 0.0263)	max_path_token_lengt h (0.0073)	entropy_url (0.1357, 0.0084)	url_length (0.0150)

TABLE 6: Top 10 Most Important Features Based on SHAP Feature Attribution

Attributions for XGBoost and the Stacking Ensemble are presented as both log-odds and their corresponding probability contributions, whereas impacts for the Random Forest and Weighted Ensemble are reported exclusively as probabilities. XGBoost, Extreme Gradient Boosting; SHAP, SHapley Additive exPlanations

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

Table 6 summarizes the top ten features identified through SHAP or SHAP-derived feature attribution across the four models, with contributions reported in the applicable output space and corresponding probability values where available. URL structural features remain dominant across the models, with `last_slash_index` ranking first in all four models. Other frequently ranked features, including `path_length`, `count_slash`, `first_dot_index`, and `has_https`, further show that characteristics of URL structure play an important role in phishing detection.

Network-level features are also present among the important features, particularly in XGBoost and the Weighted Ensemble. Features such as `dns_txt_count`, `dns_ns_count`, and `dns_a_count` appear within the top ten for XGBoost, while `dns_txt_count` also appears for the Weighted Ensemble. This suggests that DNS information contributes additional predictive information alongside the structural characteristics of URLs. Other features, including `count_special_chars`, `entropy_path`, `max_path_token_length`, and `first_slash_index`, further capture variations in URL composition that may help distinguish phishing URLs from legitimate ones.

The feature rankings are generally consistent across the four models, although their relative contributions differ. Since the models use different output spaces, the raw attribution values should not be directly compared across models. The table therefore focuses on the features that repeatedly contribute to model predictions and their relative importance within each model.

The results show that URL structure remains the most consistent source of predictive information, while DNS-related features provide additional information for the models that incorporate them. This supports the use of both URL-based and network-level characteristics in the hybrid phishing detection approach and provides evidence that the resulting predictions are based on identifiable and interpretable features.

Conclusions

This study presented an interpretable phishing website detection framework that integrates lexical and network feature extraction, ensemble ML, SHAP-based explainability, and HTTP status heuristics for inactive URL handling. RF and XGBoost were evaluated as baseline classifiers and combined using weighted soft voting and stacking ensemble learning with a logistic regression meta-learner. The evaluation demonstrated that the stacking ensemble achieved the highest overall performance among the evaluated models. An ablation study showed that lexical features provide the primary discriminatory information, while network features contribute complementary information that improves classification when integrated with lexical features. In addition, McNemar's test confirmed that the performance improvements of the stacking ensemble over the baseline classifiers are statistically significant. This study incorporated model interpretability into the phishing detection process by providing both global and local explanations of model predictions using SHAP. The explainability analysis identified URL structural characteristics and Domain Name System related attributes as the primary contributors to phishing detection and provided interpretable evidence for individual predictions and overall model behavior.

The proposed system integrates HTTP status heuristics to identify inactive URLs before ML inference and returns the corresponding HTTP status response to the user. Unlike conventional phishing detection systems that produce only binary classifications, this capability prevents unnecessary analysis of inaccessible websites while providing operational feedback for inactive URLs. In addition, SHAP-based explainability provides both global and local interpretations of model predictions without compromising the predictive performance of the stacking ensemble. This improves transparency and enables cybersecurity analysts to understand the contribution of individual features to the model's phishing classification decisions. Extraction of some network-related features remains constrained for websites protected by content delivery networks, which limits the availability of certain Domain Name System and page-level attributes. Future work can evaluate the framework on external datasets collected across different time periods, investigate additional ensemble and deep learning architectures, and examine adaptive learning strategies for evolving phishing attacks while preserving model interpretability.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Ifeoluwa D. Oseni, Ilobekemen P. Oladoja, Olumide Adewale

Acquisition, analysis, or interpretation of data: Ifeoluwa D. Oseni

Drafting of the manuscript: Ifeoluwa D. Oseni

Critical review of the manuscript for important intellectual content: Ifeoluwa D. Oseni, Ilobekemen P. Oladoja, Olumide Adewale

Supervision: Ilobekemen P. Oladoja, Olumide Adewale

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue.

Animal subjects: All authors have confirmed that this study did not involve animal subjects or tissue.

Conflicts of interest: In compliance with [the ICMJE uniform disclosure form](#), all authors declare the following:

- **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work.
- **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work.
- **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request. The data (and/or code) supporting this study are openly available on Github at <https://github.com/ifeoseni/phishing-detection-with-shap>

References

1. Aleroud A, Zhou L: [Phishing environments, techniques, and countermeasures: A survey](#). Computers & Security. 2017, 68:160-196. [10.1016/j.cose.2017.04.006](#)
2. Oyegoke TO, Amoo AO, Aigberua J: [An enhanced phishing detection system in online transactions](#). TWIST. 2024, 19:656-666.
3. Taha MA, Jabar HDA, Mohammed WK: [Machine learning algorithms for detecting phishing websites: a comparative study](#). Iraqi Journal For Computer Science and Mathematics. 2024, 5:13. [10.52866/ijcsm.2024.05.03.015](#)
4. Maneriker P, Stokes JW, Lazo EG, Carutasu D, Tajaddodianfar F, Gururajan A: [URLTran: Improving phishing URL detection using transformers](#). MILCOM 2021 IEEE Military Communications Conference. 2021, 197-204.
5. Almseidin M, Abu Zuraiq A, Al-Kasassbeh M, Alnidami N: [Phishing detection based on machine learning and feature selection methods](#). International Journal of Interactive Mobile Technologies (IJIM). 2019, 13:171-183. [10.3991/ijim.v13i12.11411](#)
6. Butnaru A, Mylonas A, Pitropakis N: [Towards lightweight URL-based phishing detection](#). Future Internet. 2021, 13:154. [10.3390/fi13060154](#)
7. Manoj P, Bhuvan Kumar Y, Rakshitha D, Megha G: [Detection and classification of phishing websites](#). Trends in Computer Science and Information Technology. 2021, 6:053-059. [10.17352/tcsit.000040](#)

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . Cureus J Comput Sci 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>

8. Piñeiro JL, Portillo LRW: [Web architecture for URL-based phishing detection based on random forest, classification trees, and support vector machine](#). *Inteligencia Artificial*. 2022, 25:107-121. [10.4114/intartif.vol25iss69pp107-121](#)
9. Calzarossa MC, Giudici P, Zieni R: [An assessment framework for explainable AI with applications to cybersecurity](#). *Artificial Intelligence Review*. 2025, 58:150. [10.1007/s10462-025-11141-w](#)
10. Shombot ES, Dusserre G, Bestak R, Ahmed NB: [An application for predicting phishing attacks: A case of implementing a support vector machine learning model](#). *Cyber Security and Applications*. 2024, 2:100036. [10.1016/j.csa.2024.100036](#)
11. Misiek M, Hyla T: [Preventing phishing attacks with browser-based URL detection](#). *Proceedings of the 58th Hawaii International Conference on System Sciences*. 2025, 58:7307-7315. [10.24251/HICSS.2025.874](#)
12. Do NQ, Selamat A, Fujita H, Krejcar O: [An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection](#). *Future Generation Computer Systems*. 2024, 161:269-285. [10.1016/j.future.2024.06.031](#)
13. Li W, Manickam S, Chong YW: [FedPhishLLM: A privacy-preserving and explainable phishing detection mechanism using federated learning and LLMs](#). *Journal of King Saud University Computer and Information Sciences*. 2025, 37:252. [10.1007/s44443-025-00267-0](#)
14. Tarapiah S, Abbas L, Mardawi O, Atalla S, Himeur Y, Mansoor W: [Evaluating the effectiveness of large language models \(LLMs\) versus machine learning \(ML\) in identifying and detecting phishing email attempts](#). *Algorithms*. 2025, 18:599. [10.3390/a18100599](#)
15. Uddin MA, Mahiuddin M, Sarker IH: [An explainable transformer-based model for phishing email detection: A large language model approach](#). *Computer Networks*. 2026, 277:112061. [10.1016/j.comnet.2026.112061](#)
16. Giudici P, Raffinetti E: [Shapley-Lorenz explainable artificial intelligence](#). *Expert Systems with Applications*. 2021, 167:114104. [10.1016/j.eswa.2020.114104](#)
17. Owa K, Adewole O: [Benchmarking machine learning techniques for phishing detection and secure URL classification](#). *International Journal of Computer Science and Mobile Computing*. 2025, 14:20-37.

How to cite this article:

Oseni I D, Oladoja I P, Adewale O (September 29, 2026) Interpretable Phishing Classification: Integrating SHapley Additive exPlanations (SHAP)-Based Explanations with Ensemble Models and Operational Status Heuristics . *Cureus J Comput Sci* 3 : es44389-026-00300-x. DOI <https://doi.org/10.7759/s44389-026-00300-x>