Security and Privacy

WILEY

RESEARCH ARTICLE OPEN ACCESS

Phishing URL Detection and Interpretability With Machine Learning: A Cross-Dataset Approach

Liyan Yi[1] | Adebayo Omotosho[2] | Hamed Balogun[3]

[1]Department of Computer Science, University of Warwick, Coventry, UK | [2]School of Business, Computing, and Social Sciences, University of Gloucestershire, Cheltenham, UK | [3]Department of Computer Science, Edge Hill University, Ormskirk, UK

Correspondence: Adebayo Omotosho (bomotosho@glos.ac.uk)

Received: 29 October 2024 | **Revised:** 15 July 2025 | **Accepted:** 14 December 2025

Keywords: datasets | explainable AI | machine learning | security | URL phishing

ABSTRACT

Phishing attacks pose a significant security threat, particularly through deceptive emails designed to trick users into clicking on malicious links, with phishing URLs often serving as the primary indicator of such attacks. This paper presents a machine learning approach for detecting phishing email attacks by analyzing the URLs embedded within these emails, using Random Forest, eXtreme Gradient Boosting, and Light Gradient Boosting Machine models. Secondary datasets are used to evaluate model behavior and examine the applicability of model features across different samples. The models are assessed using metrics such as accuracy, precision, and recall to demonstrate their effectiveness in distinguishing between benign and malicious email URLs. The SHapley Additive exPlanations (SHAP) framework is employed to interpret the models' decision-making processes and reinforce the relevance and reliability of key features. Our results show that across four test sets, the three models achieve an average classification error ≈4.03% and an average accuracy > 94%, indicating strong generalization capability across diverse datasets using the same set of features.

1 | Introduction

With the advent of the internet, email has become one of the most widely used digital communication tools in both work and daily life. However, the rise of phishing email attacks poses significant threats to network security, as attackers use social engineering techniques to deceive victims, leading to financial losses and privacy breaches for individuals and organizations alike. Phishers exploit identity theft to trick unsuspecting users into revealing sensitive information, such as account numbers and passwords [1]. Basit et al. [2] reported that phishing attacks alone cost organizations worldwide up to $9 billion. In a Kaspersky Lab's report, phishing email threats accounted for 55.97% of total traffic in 2019 [3]. In addition, phishing URL is the primary measure of reported phishing across the globe and about 1 million cases were reported in the first quarter of 2024 according to nAnti-Phishing Working Group (APWG) [4].

In recent years, Artificial Intelligence (AI), particularly machine learning and deep learning applications, has gained significant attention in the field of cybersecurity and is expected to play a key role in enhancing it. Machine learning can automatically perform specific intrusion detection tasks [5]. With the rise of phishing attacks and their evolving variants, traditional signature-based phishing defense techniques struggle to handle the diversity of these attacks [6–9]. AI's strength lies in its ability to incorporate multiscale complexity analysis, enhancing adaptive learning, reducing false positives, strengthening real-time detection, and uncovering hidden patterns in cyber threats, making it more effective against evolving and stealthy attacks [10, 11].

AI technology can significantly enhance the speed and accuracy of monitoring in email security, particularly in identifying and preventing spam and phishing emails, highlighting its potential in countering complex cyber threats [2].

Phishing detection using text-based models (e.g., BERT and deep learning) primarily focuses on analyzing email content rather than structured URL features, making these approaches computationally expensive, less interpretable, and impractical for real-time detection [12–14]. Due to the high computational resources required for inference, these models are less suited for large-scale, low-latency applications, especially when analyzing structured data like URLs, which are more efficiently handled by traditional machine learning models. Detecting phishing using structured URL-based features such as length, special character distribution, and domain attributes, leveraging tree-based models (e.g., RF and XGBoost), are efficient, interpretable, and well-suited for tabular data [15, 16]. In addition, studies have shown that high detection accuracy in AI does not always translate to good model generalization [17–19]. Most of the existing works use only a primary dataset for training and testing, without employing any secondary dataset for further model evaluation, and provide limited feature interpretation [20–25].

Hence, our approach evaluates machine learning models for phishing email detection with a focus on strong generalization and high accuracy on previously unseen URL datasets. Evaluation across diverse datasets ensures robust performance in varying contexts, mitigates overfitting to any single dataset, and demonstrates applicability suitable for real-world deployment. Accordingly, we present a phishing email detection system augmented with explainable AI techniques to interpret model decisions across heterogeneous datasets. The main contributions of this paper are:

- Development and experimental evaluation of Random Forest (RF), eXtreme Gradient Boosting (XGBoost), and Light Gradient Boosting Machine (LightGBM) models for phishing attack detection from email URLs
- Performance evaluation of the models shows that attacks can be detected with precision, recall, and accuracy (P/R/A) ≈96%/97%/97% for RF, 93%/96%/94% for XGBoost, and 95%/98%/96% for LightGBM, demonstrating the effectiveness of our models.
- Models' performance benchmarking on the secondary test sets shows that RF generalizes best overall, with the lowest average misclassification rate (≈2.90%) across all test sets, followed by XGBoost (≈4.57%) and LightGBM (≈5.42%). The average detection accuracy of all three models on all datasets is ≈96%, suggesting that the models effectively learn from the features and exhibit strong generalization capability.
- Explanation of model behavior using SHAP to interpret the impact of top features. Results show that features such as *google_index* and *page_rank* are consistently influential across models, reinforcing their reliability and the models' suitability for real-world deployment.
- Design and deployment of models on the web using the Flask framework, which provides an easy-to-use user interface for phishing email detection.

1.1 | Research Questions

Our research focuses on evaluating the balance between model accuracy and generalization, particularly when applied to new, unseen datasets. The following research questions guide our investigation:

- **RQ1.** *Can we obtain significant performance scores for phishing URL detection across multiple datasets?*
- **RQ2.** *What is the impact of the key features used by the models?*

The remainder of this paper is organized as follows: Section 2 discusses related work on email phishing and compares our work to recent research; Section 3 describes the design and implementation strategies employed in this paper; Section 4 presents the evaluation results and their discussion; and Section 5 provides a summary of this paper.

2 | Related Work

There are many studies that show machine learning can be used for email phishing detection. This section presents a summary of those relevant to our research, highlighting how they differ from the approach taken in this paper. Sundararaj and Kul [20] reported that the use of Support Vector Machine (SVM), Logistic Regression (LR), and Decision Tree (DT) models in their study, which were trained and tested on a dataset of 525 754 samples. The results show that these models achieved precision and accuracy rates > 80%, with a recall rate > 60%. Our paper is distinct in that it focuses on URL data, rather than text data, by optimizing URL characteristics to address phishing detection. Fette et al. [26] paper is based on the comparison of training and testing of multiple machine learning models on a highly unbalanced 7810 sample dataset. Their results showed that the random forest model had the highest accuracy, reaching 99.5%. However, the high accuracy achieved by Fette et al. [26] is on the same dataset, but the accuracy obtained in our paper is from validation performed over multiple datasets to demonstrate our models consistency. Fang et al. [27] used an email dataset containing 8780 samples to evaluate the performance of the thymocyte-expressed molecule involved in selection (THEMIS), Long Short-Term Memory (LSTM), and convolutional neural networks (CNN) models for email classification tasks. THEMIS achieved 99.664% precision, 99% recall, and 99.848% accuracy, LSTM achieved 93.258% precision, 83% recall, and 97.38% accuracy, and CNN had 85.473% precision, 84.333% recall, and 96.583% accuracy. Our paper is different in that we experimented with a larger number of datasets, and our total training and evaluation dataset contains 46 351 samples. Gualberto et al. [28] utilized email text of 6429 samples and using feature sets of various sizes while applying them to support vector classifier (SVC), Naive Bayes (NB), LR, k-Nearest Neighbors (KNN), DT, RF, XGBoost, and Multilayer Perceptron (MLP) models. Four feature selection methods were used: Principal Component Analysis (PCA), Latent Semantic Analysis (LSA), mutual information, and Chi-square test. RF achieved 100% precision, accuracy, and recall with the Chi-square test on 100 features, while XGBoost achieved 100% precision, accuracy, and recall across LSA-feature

set with 25 features. Unlike this paper, our experiment focuses on the features of URLs within emails. Akinyelu et al. [29] developed a phishing detection model based on 2000 text email samples. Using RF, the model achieved 99.75% accuracy, 99.47% precision, and 97.50% recall. However, the results obtained is based on a small training and testing dataset. In our paper, training is conducted on 11 430 samples and testing is performed on a total of 37 207 samples from different sources. Abu-Nimeh et al. [30] compared the performance of six models: LR, Classification And Regression Tree (CART), Bayesian Additive Regression Tree (BART), SVM, RF, and Neural Networks (NN) in predicting phishing emails. Using a dataset of 2889 samples, the study evaluated these models through 10 cross-validations, with LR achieved 95.11% precision and 82.96% recall. This paper is also distinct because the experiment is performed on a total dataset of 2889 samples, with the evaluation limited to a subset of this sample. Zhan et al. [21] used Naive Bayes model to train and test machine learning models on 1000 data samples. Feature extraction is performed using Semantic Latent Word Embeddings (SLWE) and maximum likelihood estimation (MLE). The results showed that the recall for Naive Bayes combined with SLWE was 98.9%, compared to 98.7% for Naive Bayes based on MLE. This paper also has a more limited evaluation compared to our work. While text-based analysis is useful for detecting social engineering attacks, it requires significant computational resources due to the large size of annotated email corpora and the complexity of text processing models, making them computationally expensive [31], URL-based detection is often more effective for phishing detection in large-scale automated defenses and real-time threat prevention [23] (Table 1). Additionally, URL detection is language-independent, enabling it to effectively identify phishing attempts across different languages and attack vectors, whereas text-based approaches can struggle with multilingual detection [40]. However, this does not guarantee that the URL-based approach will always outperform the text-based approach in terms of detection accuracy. Table 2 presents some detection performance results of text-based approaches.

On the use of explainable AI (XAI) in phishing detection, Shafin [47] discussed the application of XAI methods SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME) in phishing detection, focusing on generating feature explanations for the RF classifier to identify key predictors. SHAP offers global feature importance by revealing the overall impact of each feature on model predictions, while LIME provides additional local explanations for individual instances. Together, SHAP and LIME enhance the robustness of feature selection and improve model interpretability. Hernandes et al. [48] also reiterated LIME's ability to generate local explanations by creating perturbed samples near the input data, thus revealing the basis of predictions in black-box models (e.g., Random Forest and Support Vector Machine) for specific cases. LIME uses color distinctions to illustrate the varying impacts of features on legitimate URLs versus phishing URLs. Additionally, explainable boosting machine (EBM), a transparent white-box model, employs an additive framework to generate predictions, offering both global feature importance and local explanations for individual cases. This high level of transparency makes EBM suitable for phishing detection applications that prioritize interpretability, though sometimes at the expense of performance. Puri et al. [49] emphasized the use of the Tree SHAP method to interpret feature contributions within the CatBoost model for phishing detection, emphasizing the influence of critical features (such as URL_of_Anchor and SSLfinal_State) on prediction outcomes. This approach provides both global and local explanations, enhancing comprehension of the model's decision-making process. Greco et al. [50] also explored the application of XAI with SHAP analysis to reveal the nonlinear effects of behavioral habits (e.g., checkhttps and clickwocheck) on phishing susceptibility. By combining deep neural networks with SHAP values,

TABLE 1 | Comparison of our work to other URL classifiers.

Author	Model	Sample size	Precision	Recall	Accuracy	Benchmark
This paper	**RF, LightGBM, and RF**	**11 430**	**94.7%**	**97.0%**	**95.6%**	**Yes**
Coyotes et al. [32]	CNN, RNN, and MLP	18 778	NA	NA	94.15%	No
Hiransha et al. [33]	CNN	19 778	NA	NA	95.5%	No
Jain et al. [34]	RF	4056	98%	99%	99%	No
Bagui et al. [35]	LSTM and CNN	18 366	NA	NA	96.07%	No
Shalini et al. [36]	RF and ANN	5574	NA	NA	99.25%	No
Unnithan et al. [37]	DT, NB, AdaBoost, LR, K-NN, SVM, and RF	14 866	88.95%	100%	88.08%	No
Bhatti et al. [38]	LSTM	42 533	NA	NA	97%.	No
Ra et al. [39]	CNN, RNN, LSTM, and MLP	18 778	94.67%	89.33%	86.33%	No
Dewis et al. [24]	LSTM and MLP	20 324	96.5%	NA	96.5%	No
Chen and Chen [25]	XGBoost, RF, SVM, LR, KNN, and ELM	28 321	NA	NA	97.6%	No
Sahingoz et al. [23]	DT, Adaboost, RF, and NB	73 575	94.55%	91.5%	96.02%	No
Rao and Pais [22]	J48, RF, SMO, LR, MLP, BN, SVM, and AM1	3526	99.40%	NA	99.24%	No

TABLE 2 | Comparison of text-based classifiers.

Author	Model	Sample size	Precision	Recall	Accuracy	Benchmark
Fette et al. [26]	SVM	7810	NA	NA	99%	No
Sundararaj and Kul [20]	SVM, RF, and LG	525 754	93%	70%	90%	No
Gualber et al. [28]	SVM, NB, LR, KNN, DT, RF, XGBoost, and MLP	6429	99%	99%	99%	No
Akinyelu and Adewumi [29]	RF	2000	99.47%	97.5%	NA	No
Abu-Nimeh et al. [30]	RF	2889	93%	88.88%	NA%	No
Somesha and Pais [41]	RF, DT, XGBoost, and LR	60 956	NA	NA	99%	No
Atawneh and Aljehani [42]	CNN and LSTM	7810	98%	98%	98%	No
Altwaijry et al. [43]	CNN	6428	100%	99.32%	99.68%.	No
Hiransha et al. [33]	CNN	4583	NA	NA	96%	No
Fang et al. [27]	LSTM and CNN	8780	89%	NA	96.5%	No
Yasin and Abuhasan [44]	RF and J48	10 538	98.5%	98.5%	99%	No
Butt et al. [45]	SVM, NB, and LSTM	4000	97.7%	99%	98.2%	No
Alotaibi et al. [46]	CNN	6428	98.80%	99.54%	99.42%	No

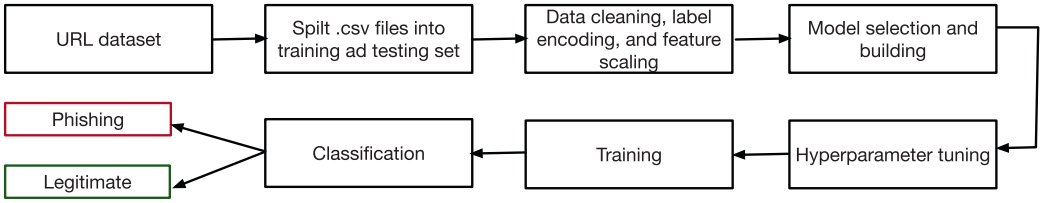

FIGURE 1 | Experimental workflow.

the study identifies key behavioral features and provides personalized anti-phishing recommendations. Partial dependence plots and waterfall charts further detail individual phishing risks, supporting targeted anti-phishing training and underscoring the significance of behavioral factors in phishing vulnerability.

All of these studies used only the primary data set for training and testing, and did not use any secondary dataset for further evaluation of the model. Additionally, Table 1 shows the comparison of our paper to similar URL based works. However, in our multi datasets predictor, we employ the use of the SHAP framework because it is one of the most popular XAI models, particularly for its versatility and interpretability in machine learning [51].

3 | Design and Implementation

3.1 | Approach Overview

The complete workflow of our experiment, from the processing of the raw URL dataset to the classification task is shown in Figure 1. First, the data set is divided into a training set and a test set, followed by data cleaning, label coding, and feature scaling pre-processing steps. The model is then trained and hyperparameter-tuned to optimize model performance. After the training is complete, the models' performance is evaluated on the test dataset to accurately distinguish between phishing and legitimate URLs. Algorithm 1 summarizes the algorithmic steps taken to complete the detection task and benchmark the models on heterogeneous datasets, $\beta_1 \ldots \beta_3$, and illustrates how we derived the statistics for interpreting the impact of the features. The description of how these are accomplished is provided in the following sections.

3.1.1 | Development and Deployment Environment

Development is carried out using the Python programming language in the PyCharm IDE, which offers many developer-friendly features such as code completion, project management, powerful debugging tools, and extensive support for the Python ecosystem. Also, Anaconda is used for managing the Python environment. After evaluation, the model with the best performance is deployed and integrated into an API that runs on end users' machines. Flask, a lightweight Python web framework, is used for model deployment, making it easy to encapsulate trained models into back-end services. Two API endpoints are developed using Flask: */predict*, which processes prediction requests for a single data point, and */predict-file*, which accepts CSV files, performs batch predictions, and returns prediction results for datasets, thus accommodating data processing at different scales. The front-end interface is built with HTML, CSS, and JavaScript, providing a user-friendly, interactive interface for users to communicate with the back-end model's APIs. Additionally, a desktop application is built using Python's standard GUI library, Tkinter.

ALGORITHM 1 | Phishing URL detection and SHAP analysis.

Input: Dataset β, Test dataset fraction $p = 0.2$, Benchmark datasets $\beta_{test1}, \beta_{test2}, \beta_{test3}$
Output: Predicted labels, SHAP values for top features

def `PreprocessData`(β, p)**:**
  Split β into training set β_{train} and test set β_{test} where $\beta_{test} = p \cdot \beta$;
  return $\beta_{train}, \beta_{test}$;
def `TrainModel`(β_{train})**:**
  model ← Initialize model ;
  model.fit(β_{train});
  return model;
def `EvaluateModel`(*model,* β_{test})**:**
  predictions ← model.predict(β_{test});
  eval_metrics ← Compute evaluation metrics;
  return accuracy, predictions;
def `GetTopFeatures`(*model, n*)**:**
  top_features ← Select top n features from model based on feature importance;
  return top_features;
def `ComputeSHAP`(*model, β_test, top_features*)**:**
  explainer ← SHAP explainer initialized with model;
  shap_values ← explainer.shap_values($\beta_{test}, top\_features$);
  return shap_values;
$\beta_{train}, \beta_{test}$ ← `PreprocessData`(β, p);
model ← `TrainModel`(β_{train});
eval_metrics, predictions ← `EvaluateModel`(*model,* β_{test});
foreach $\beta_{test} \in \{\beta_{test1}, \beta_{test2}, \beta_{test3}\}$ **do**
  labels ← `EvaluateModel`(*model,* β_{test});
top_features ← `GetTopFeatures`(*model, n*);
shap_values ← `ComputeSHAP`(*model, β_test, top_features*);
return eval_metrics, predictions, labels, shap_values;

3.2 | Data Collection

This paper uses datasets obtained from Kaggle, a public source of machine learning datasets that has been used in previous research. Our training dataset consists of 11 430 URL phishing samples with 87 features, of which the legitimate and phishing samples are 5715 each. The features in the dataset are shown in Table 3. Datasets from Tiwari [52], Winson [53], and Manish [54] are used to benchmark our models' detection performance. The benchmark data set provides our approach with diverse samples to assess its generalization capability. Table 4 summarizes the benchmark datasets, which have also been used in other research papers for URL-based phishing detection [55–57].

During the data preprocessing phase, several challenges were encountered, particularly in handling missing values in datasets from different sources and ensuring data consistency and formatting across these datasets. To address these issues, two main methods were adopted: the padding method and manual review.

- Manual review: Manual review is conducted to identify and address inaccuracies, duplicates, or extraneous information in the data. This process further improves data quality, ensuring that the data used for training models is as accurate as possible.
- Padding method: For missing URL headers, a simple and effective approach is taken: automatically adding "http://" or "https://" before the missing URL. This step ensures consistency across all URL formats and reduces errors during model processing. Additionally, for missing non-sequential features, the padding method is used to fill missing values with 0 or an empty string, ensuring the data's integrity and consistency.

TABLE 3 | Features.

Data type	Features
URL	url, length_url, length_hostname, ip, nb_dots, nb_hyphens, nb_at, nb_qm, nb_and, nb_or, nb_eq, nb_underscore, nb_tilde, nb_percent, nb_slash, nb_star, nb_colon, nb_comma, nb_semicolumn, nb_dollar, nb_space, nb_www, nb_com, nb_dslash, http_in_path, https_token, ratio_digits_url, ratio_digits_host, punycode, port, tld_in_path, tld_in_subdomain, abnormal_subdomain, nb_subdomains, prefix_suffix, random_domain, shortening_service, path_extension, nb_redirection, nb_external_redirection, length_words_raw, char_repeat, shortest_words_raw, shortest_word_host, shortest_word_path, longest_words_raw, longest_word_host, longest_word_path, avg_words_raw, avg_word_host, avg_word_path, phish_hints, domain_in_brand, brand_in_subdomain, brand_in_path, suspecious_tld, statistical_report, nb_hyperlinks, ratio_intHyperlinks, ratio_extHyperlinks, ratio_nullHyperlinks, nb_extCSS, ratio_intRedirection, ratio_extRedirection, ratio_intErrors, ratio_extErrors, login_form, external_favicon, links_in_tags, submit_email, ratio_intMedia, ratio_extMedia, sfh, iframe, popup_window, safe_anchor, onmouseover, right_clic, empty_title, domain_in_title, domain_with_copyright, whois_registered_domain, domain_registration_length, domain_age, web_traffic, dns_record, google_index, page_rank

TABLE 4 | Benchmark datasets.

Dataset	Source	Data type	Phishing	Legitimate	Total sample
Winson [53]	Kaggle	URL	9715	9716	19 431
Manish [54]	Kaggle	URL	5741	5740	11 481
Tiwari [52]	Kaggle	URL	4009	0	4009

3.3 | Evaluation Method

The performance of the developed models are evaluated using key metrics such as accuracy, recall, F1 score, confusion matrix, cross-validation, and Area Under the ROC Curve (AUC), which not only reflect the predictive power of the model but also provide transparency into the model's decision-making process. Equations (1–6) explain each metric. In these equations, a true positive (TP) is when the model correctly identifies a phishing email as phishing, representing a successful detection of a malicious email. A true negative (TN) is when the model correctly identifies a legitimate email as non-phishing, indicating an accurate classification of a safe email. A false positive (FP) is when the model incorrectly classifies a legitimate email as phishing, resulting in a harmless email being flagged as malicious. Finally, a false negative (FN) is when the model incorrectly identifies a phishing email as legitimate, meaning a malicious email goes undetected, which poses a risk to users. Additionally, the SHAP XAI method is used to explain the predictions of our machine learning models using the formula in Equation (7).

3.3.1 | Accuracy

This is the percentage of emails correctly identified as either phishing or legitimate out of the total emails analyzed. It reflects the model's ability to accurately differentiate between phishing and non-phishing emails.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

3.3.2 | Precision

This is the percentage of emails identified as phishing that are actually phishing. It measures the model's ability to avoid falsely labeling legitimate emails as phishing.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

3.3.3 | Recall

This is also known as the true positive rate (TPR); it is the percentage of actual phishing emails that are correctly identified by the model. It reflects the model's ability to catch as many phishing emails as possible without missing them.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

3.3.4 | F1-Score

This is the harmonic mean of precision and recall. It provides a balanced measure of the model's ability to both accurately identify phishing emails and minimize missed detections, especially when there is an imbalance between phishing and legitimate emails.

$$\text{F1 score} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4}$$

3.3.5 | AUC

AUC measures the model's ability to distinguish between phishing and legitimate emails across various threshold settings (x). A higher AUC indicates better performance in separating phishing emails from non-phishing ones, regardless of the decision threshold. AUC is computed by plotting TPR against the false positive rate (FPR)—the percentage of legitimate emails incorrectly classified as phishing out of all legitimate emails evaluated.

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

$$\text{AUC} = \int_0^1 TPR(x)dFPR(x) \tag{6}$$

3.4 | SHAP

The SHAP values help to interpret how different features can influence the classification. Higher positive SHAP values indicate features that strongly contribute to classifying an email URL as phishing, while negative values can indicate features that suggest the email is legitimate.

$$\phi_i = f(X) - f(X\backslash\{i\}) \tag{7}$$

where ϕ_i is the SHAP value for feature i, $f(X)$ is the model's predicted output when all features are included, $f(X\backslash\{i\})$ is the model's predicted output when feature i is excluded from the feature set X.

3.5 | Model Selection and Training

Following the no free lunch machine learning theorem [58], seven algorithms are initially evaluated: Support Vector Machines, XGBoost, Naive Bayes (NB), Random Forest (RF), Logistic Regression (LR), Light Gradient Boosting (LightGBM), Categorical Boosting (CatBoost), and Gradient Boosting (GB). However, only three algorithms—RF, XGBoost, and LightGBM—are selected for the remaining part of the experiments in this paper. These algorithms are chosen because they demonstrated relatively high accuracy and fit our datasets better. RF and XGBoost excel due to ensemble learning. RF reduces overfitting by averaging multiple decision trees, while XGBoost improves accuracy by sequentially correcting errors using gradient boosting. LightGBM, also a gradient boosting algorithm, is optimized for speed and memory efficiency, making it particularly suitable for large-scale, high-dimensional data. All three algorithms balance accuracy, performance, and scalability, which makes them highly effective for phishing URL detection. In the preliminary results shown in Table 5, only RF clearly stands out among the seven models.

TABLE 5 | Models selection.

Model	Accuracy
SVM	0.94
XGBoost	0.95
NB	0.67
RF	0.97
LR	0.94
LightGBM	0.96
CatBoost	0.95

3.5.1 | Model Development

For model training, the pre-processed dataset is divided into two parts: an 80% training set and a 20% test set. The K-fold cross-validation method is used to optimize the model parameters to ensure that the model has good generalization ability across different data subsets. Model parameters are tuned using grid search, adjusting and optimizing the parameter settings according to the characteristics of different algorithms. We use *GridSearchCV* from *scikit-learn* for hyperparameter optimization. The optimal metrics for our models are shown in Table 6.

The key performance indicators required for evaluating the efficiency and performance of the model training, such as accuracy, recall, F1 score, and AUC, are recorded during the training process. The training performance of our three best models with the lowest FN is presented in Table 7 (Figure 2). RF has 1.04% FP and 1.32% FN, XGBoost has 2.29% FP and 2.65% FN, and LightGBM has 0.19% FP and 0.23% FN. The confusion matrices (CM) in Figures 3–5 show the mispredictions made by each model during training where 1 represents phishing emails and 0 represents legitimate emails. The ROC curves for RF (AUC ≈ 0.997), XGBoost (AUC ≈ 0.988), LightGBM (AUC ≈ 0.996), in Figure 2,

TABLE 6 | Hyperparameters from *GridSearchCV*.

Algorithm	Hyperparameters	Meanings	Optimal values
XGBoost	*n* estimators	Number of trees	100
	Maximum depth	Maximum depth of tree	2
	Learning rate	Shrinkage coefficient of tree	0.1
RF	*n* estimators	Number of trees in forest	200
	Maximum depth	Maximum depth of a tree	9
	Minimum split	Minimum samples of split for nodes	2
	Minimum leaf	Minimum sample of nodes for leaf	1
LightGBM	*n* estimators	Number of trees	200
	Maximum depth	Maximum depth of tree	5
	Learning rate	Shrinkage coefficient of tree	0.1
	Colsample_bytree	Fraction of features randomly chosen per tree	0.6
	Subsample	Fraction of training data used per tree	0.6

TABLE 7 | Models training performance.

Model	Precision	Recall	Accuracy	F1-Score	AUC
RF	0.979	0.973	0.976	0.976	0.997
XGBoost	0.954	0.947	0.950	0.950	0.988
LightGBM	0.996	0.995	0.995	0.995	0.996

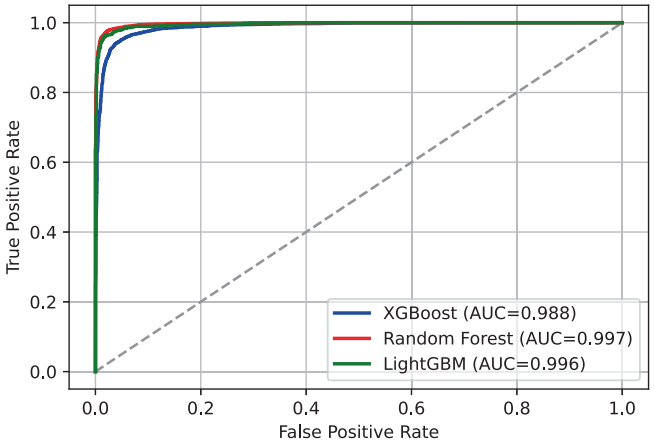

FIGURE 2 | ROC curve.

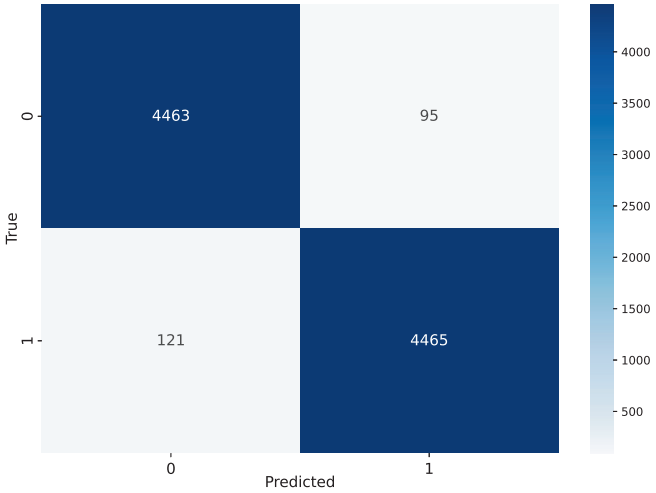

FIGURE 3 | RF confusion matrix.

reflect each model's ability to balance TPR and FPR in detecting phishing URLs. The high AUC values indicate that the models are effective in distinguishing between legitimate and phishing URLs, with very few false positives. Additionally, the ROC curves remain far from the no-skill classifier (diagonal) line, demonstrating that both models make significantly better-than-random predictions. RF slightly outperforms XGBoost, suggesting it provides a better trade-off between correctly identifying phishing URLs and minimizing false alarms. This is very important because accuracy and efficiency are crucial for maintaining system performance while preventing phishing attacks.

4 | Discussion

This section is discussed based on our two research questions.

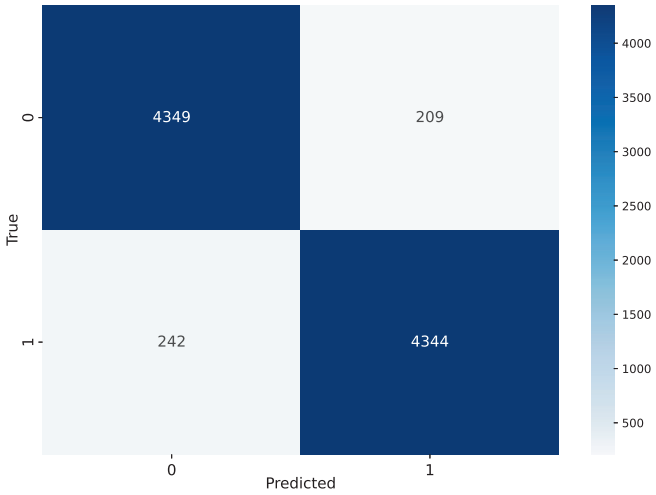

FIGURE 4 | XGBoost confusion matrix.

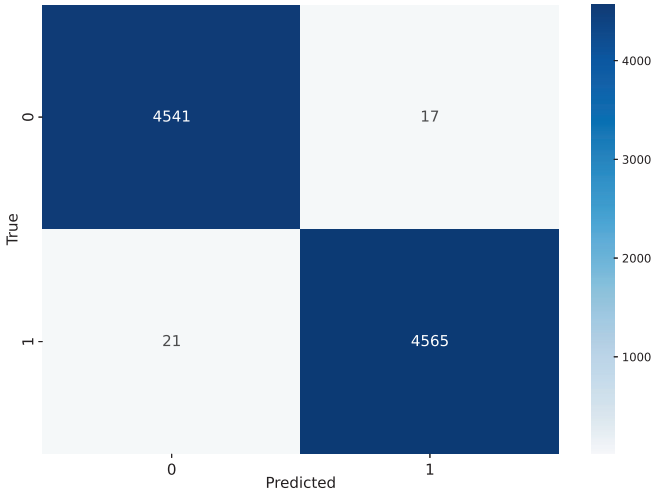

FIGURE 5 | LightGBM confusion matrix.

4.1 | RQ1: Can We Obtain Significant Performance Scores for Phishing URL Detection Across Multiple Datasets?

In this phase, the same preprocessing steps applied to the training dataset are also applied to the test dataset. The trained models are then used to make predictions, while recording the evaluation metrics. The evaluation datasets are denoted as B1 … B4, where B1 represents the 20% test set (*samples from the original data*), B2 and B3 represent the full datasets from Winson [53] and Manish [54], and B4 is a subset of Tiwari [52]. The B1 … B3 evaluation datasets contain both legitimate and phishing samples, whereas the B4 dataset consists of only one class (phishing). The B4 dataset lacks class diversity and is not suitable for full comparative evaluation across classification metrics. However, we use a subset of 4009 samples to assess our models' behavior on phishing instances in isolation and to provide insights into the false negative rate. Additionally, the subset is selected to ensure a manageable and balanced evaluation scope, consistent with the size of the other test sets. Using the full dataset can disproportionately emphasized one class, potentially skewing the interpretation of performance metrics.

TABLE 8 | Performance of models on URL test dataset B1.

Model	Precision	Recall	Accuracy	F1-score
RF	0.974	0.970	0.972	0.972
XGBoost	0.962	0.953	0.958	0.957
LightGBM	0.969	0.966	0.968	0.968

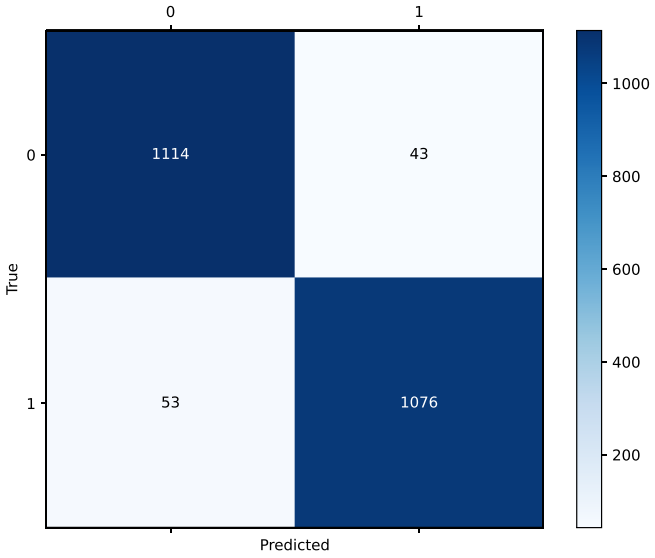

FIGURE 6 | RF CM — B1.

The stage is divided into two phases, which are discussed in the following sections.

4.1.1 | Phase 1 — Using Test Set

Using the B1 test set which contains 2286 samples, Table 8 shows the performance scores of our models. Figures 6–8 show the number of mispredictions made by the models. We have FN of ≈2.32% (53 samples) and FP of ≈1.88% (43 samples) for RF; FN ≈ 2.58% (59 samples) and FP ≈ 2.10% (48 samples) for XGBoost; FN ≈ 1.66% (38 samples) and FP ≈ 1.49% (34 samples) for LightGBM. The average detection accuracy of the models is ≈97%, indicating that they have strong classification capabilities.

4.1.2 | Phase 2 — Using Benchmark Set

This phase tests the model's generalization capability on the benchmark datasets B2 . . . B4. The dataset B2 contained 19 431 samples, each with 86 features. There are two categories of data, 9716 legitimate and 9715 phishing. Table 9 and Figures 9–11 display the performance scores of the phishing email classification made by the models on the URL dataset B2. The misclassification rates (FN/FP) are 1.64%/2.86% for RF, 1.62%/6.25% for XGBoost, and 0.55%/6.42% for LightGBM.

The dataset B3 has 11 481 items including 5741 legitimate items and 5740 phishing items with 87 features. Table 10 and Figures 12–14 display the performance scores of the phishing email classification made by the three models on the URL dataset B3. The misclassification rates (FN/FP) are 1.49%/1.25% for RF, 2.54%/2.32% for XGBoost, and 0.46%/0.45% for LightGBM.

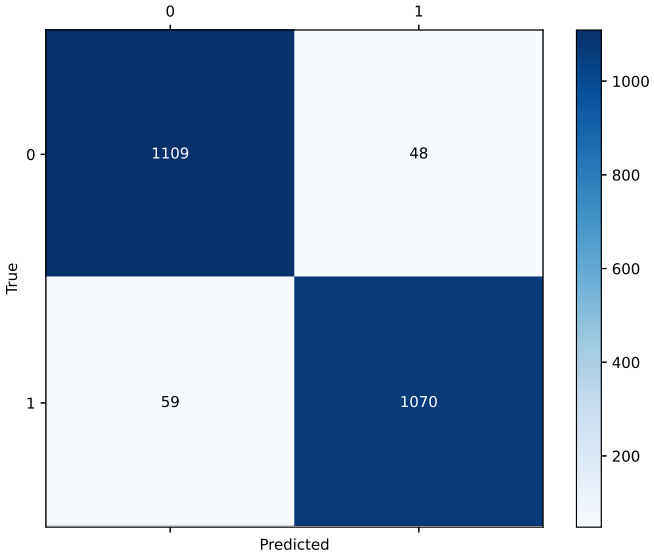

FIGURE 7 | XGBoost CM — B1.

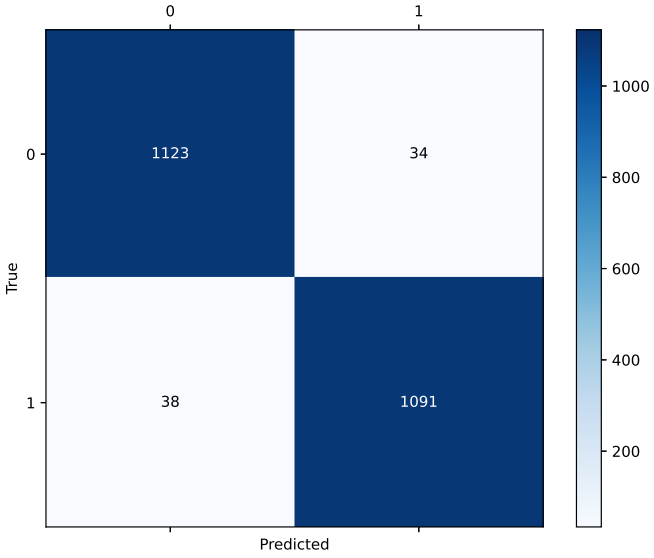

FIGURE 8 | LightGBM CM — B1.

TABLE 9 | Performance of models on URL dataset B2.

Model	Precision	Recall	Accuracy	F1-score
RF	0.944	0.967	0.954	0.955
XGBoost	0.886	0.967	0.921	0.924
LightGBM	0.885	0.989	0.930	0.934

The URL dataset B4 consists entirely of phishing URLs and we randomly selected 4009 samples. Table 11 and Figures 15–17 display the performance scores of the phishing email classification made by the three models on the URL dataset B4. The model's misclassification rates in terms of FN is 0.18%, for RF, 0.85%, for XGBoost and 11.35%, for LightGBM Figures 15–17 have zeros for the actual negative class, as it does not exist in this dataset. Therefore, the confusion matrices are included to maintain consistency

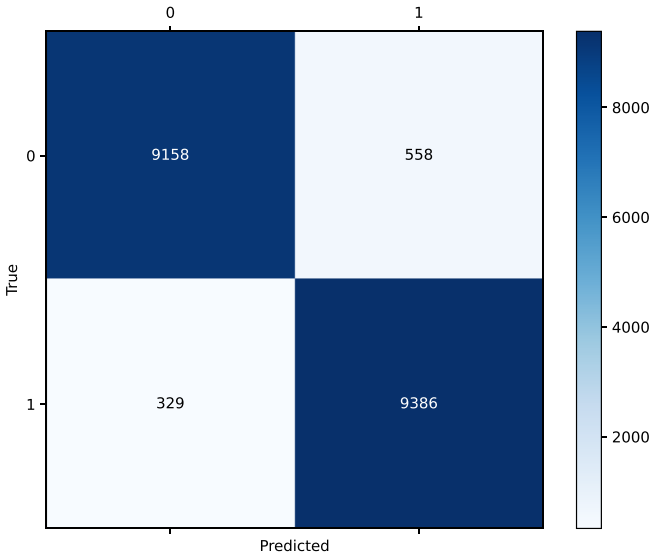

FIGURE 9 | RF CM — B2.

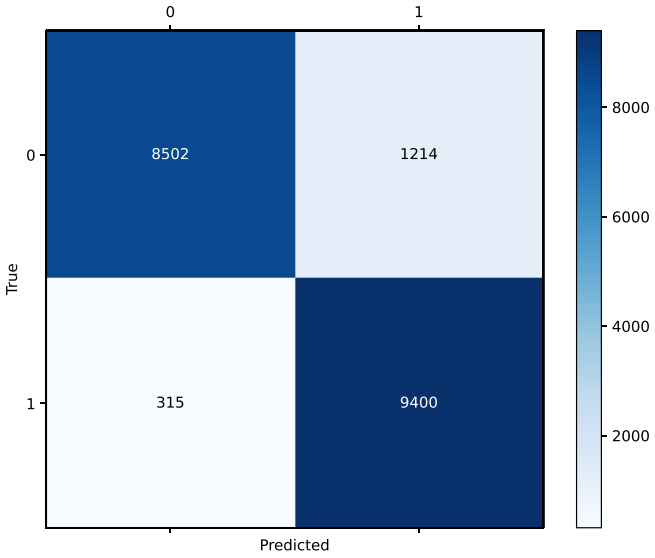

FIGURE 10 | XGBoost CM — B2.

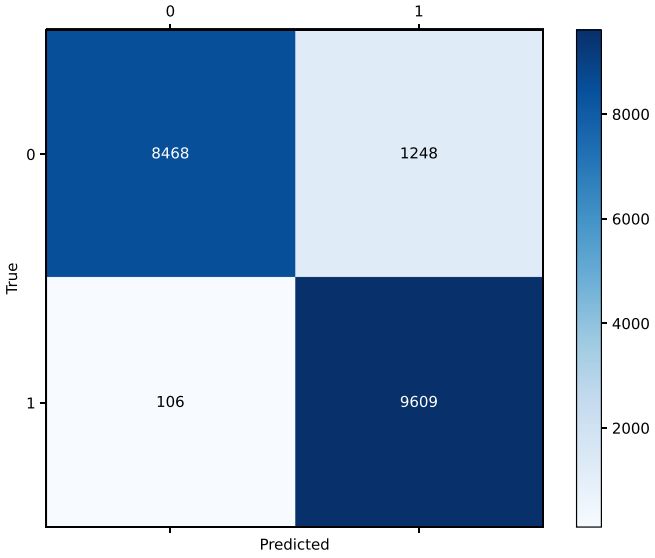

FIGURE 11 | LightGBM CM — B2.

TABLE 10 | Performance of models on URL dataset B3.

Model	Precision	Recall	Accuracy	F1-score
RF	0.974	0.970	0.972	0.972
XGBoost	0.953	0.949	0.951	0.951
LightGBM	0.990	0.990	0.990	0.990

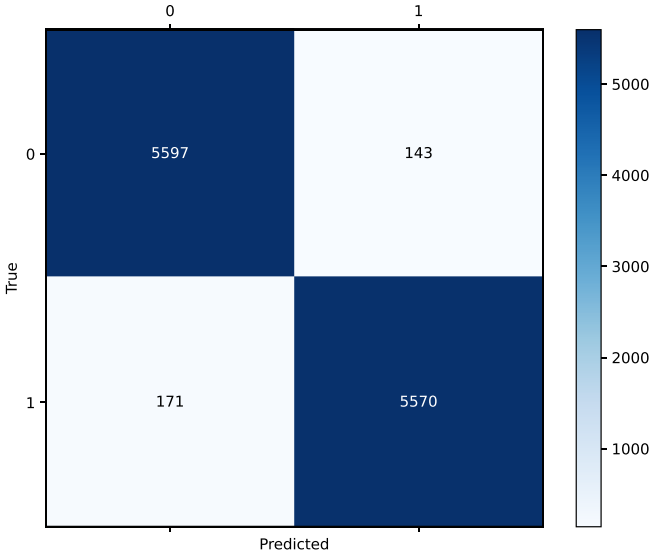

FIGURE 12 | RF CM — B3.

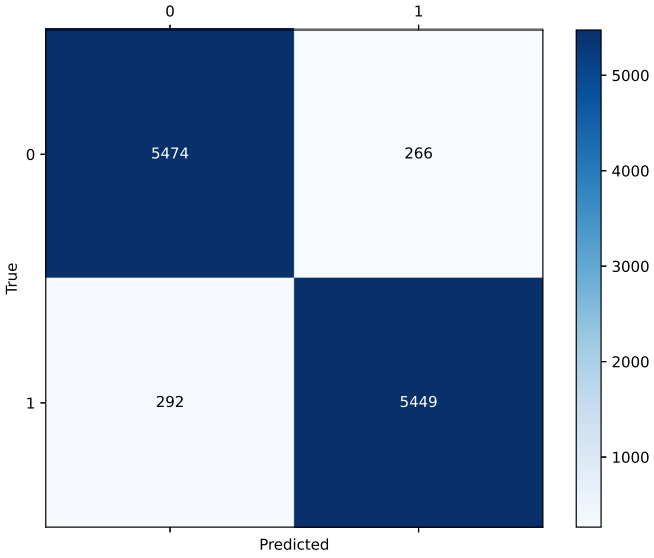

FIGURE 13 | XGBoost CM — B3.

in our presentation. Nevertheless, our classification results show TP detection rates of 99.83% for RF, 99.15% for XGBoost, and 88.65% for LightGBM.

RF generalizes best overall, as it has the lowest average misclassification rate (≈2.90%) across all test sets, followed by XGBoost (≈4.57%) and LightGBM (≈5.42%). The average detection accuracy of all three models on the benchmark datasets is ˜96.42%, suggesting that the models effectively learn from the features and exhibit strong generalization capability.

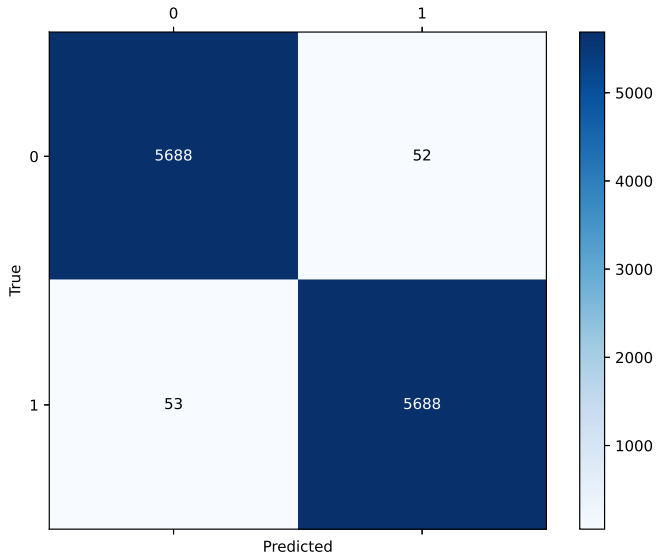

FIGURE 14 | LightGBM CM — B3.

TABLE 11 | Performance of models on URL dataset B4.

Model	Precision	Recall	Accuracy	F1-score
RF	1.000	0.998	0.998	0.999
XGBoost	1.000	0.991	0.991	0.995
LightGBM	1.000	0.886	0.886	0.939

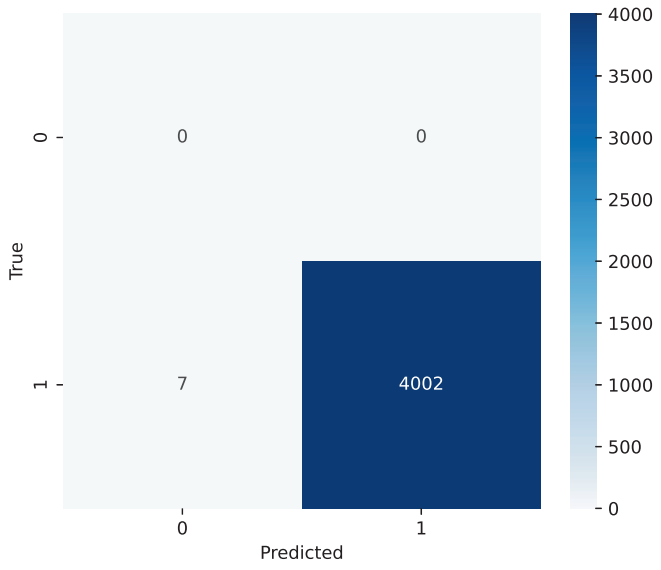

FIGURE 15 | RF CM — B4.

4.2 | RQ2: What Is the Impact of the Key Features Used by the Models?

In this section, we focus only on the impact and interpretation of the top features used by RF, XGBoost, and LightGBM. To analyze the important features, we begin with the traditional *feature importance* for RF, XGBoost, and LightGBM shown in Figures 18–20, respectively. They show how much each feature contributes to our models' predictions.

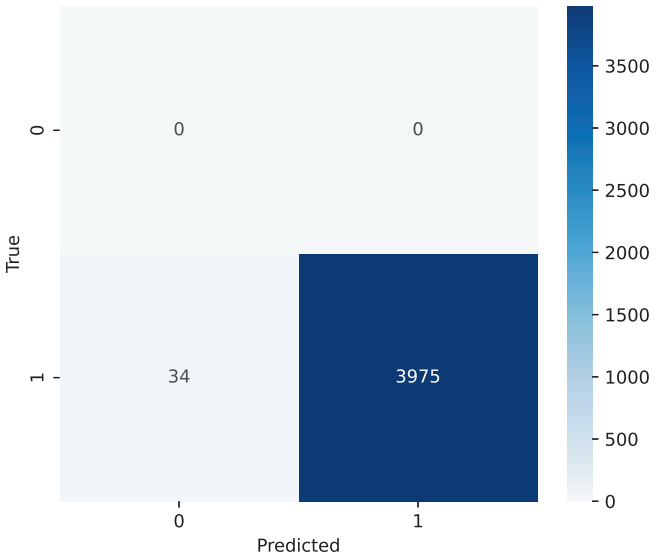

FIGURE 16 | XGBoost CM — B4.

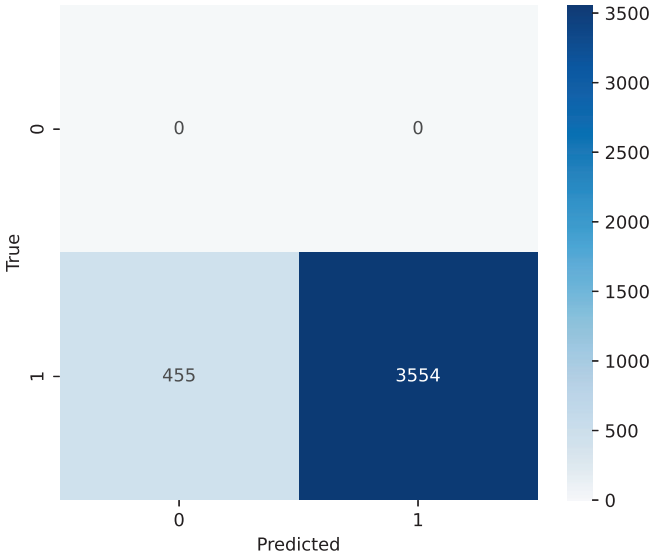

FIGURE 17 | LightGBM CM — B4.

In descending order of importance, the top five features identified by the RF model are *google_index*, *page_rank*, *nb_hyperlinks*, *web_traffic*, and *nb_www*. These features highlight the model's emphasis on web reputation and structural characteristics. *google_index* and *page_rank* capture the trust and authority of a webpage based on its visibility and ranking in search engines. *nb_hyperlinks* provides insight into the content richness or potential redirection behavior of a page. Meanwhile, *web_traffic* reflects the site's popularity, and *nb_www* helps in identifying anomalies in domain structure often found in phishing URLs. Together, these features enable effective detection of malicious websites. This is the same for XGBoost but the descending order of importance is *google_index*, *nb_hyperlinks*, *nb_www*, *page_rank*, and *web_traffic*.

For the LightGBM model, the top five features in descending order of importance are *domain_age*, *nb_hyperlinks*, *page_rank*, *domain_registration_length*, and *length_hostname*. These

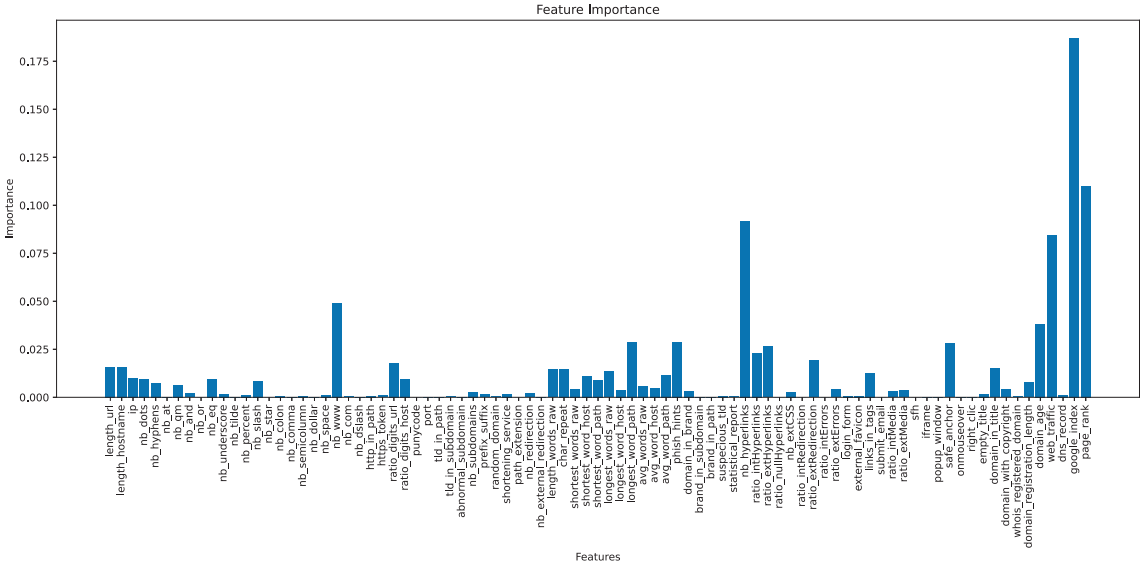

FIGURE 18 | RF feature importance.

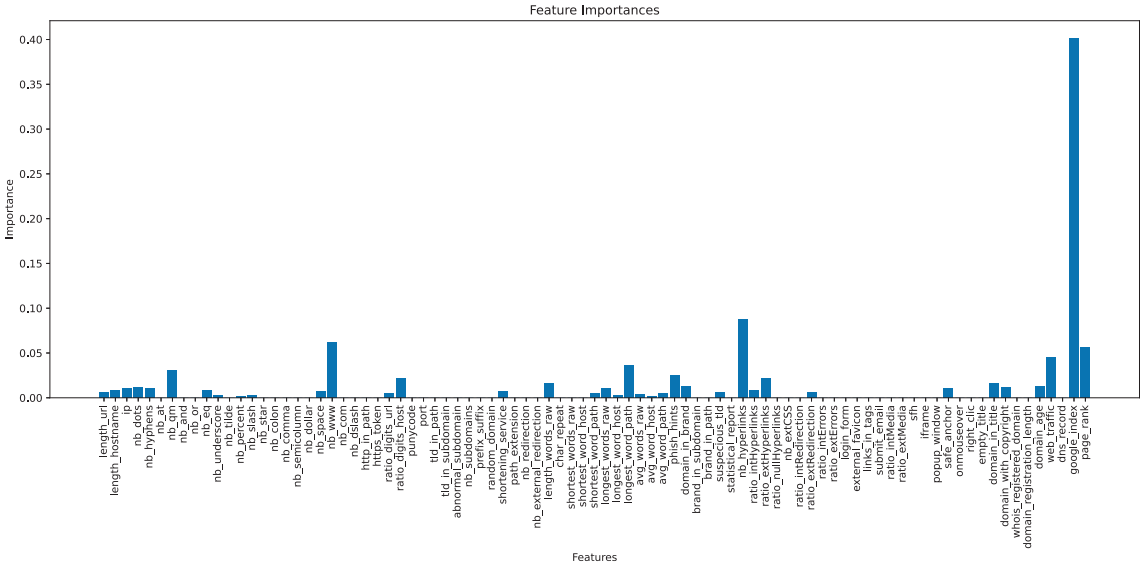

FIGURE 19 | XGBoost feature importance.

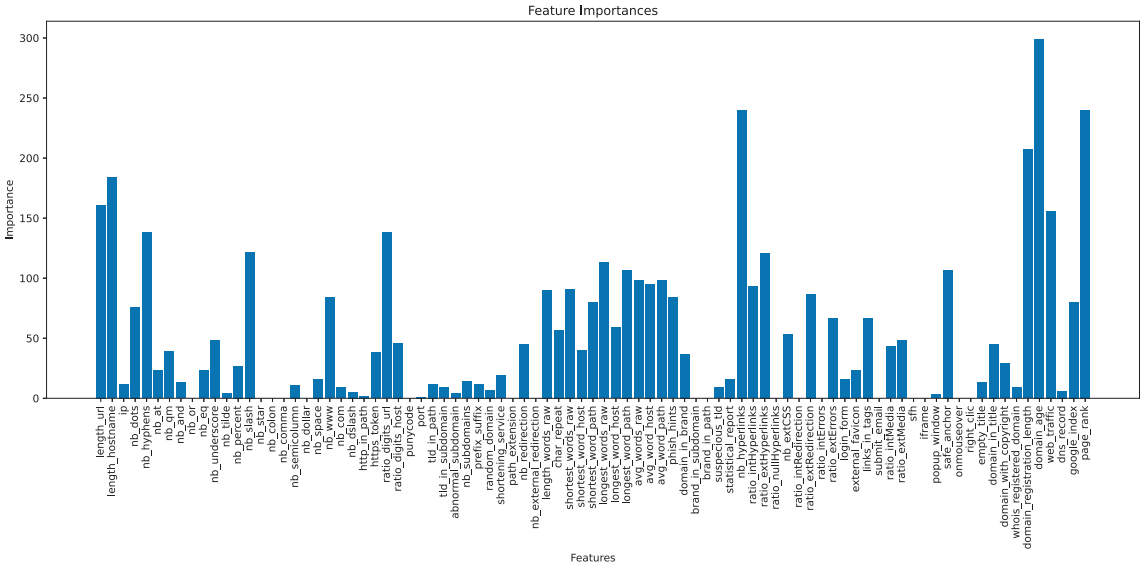

FIGURE 20 | LightGBM feature importance.

features reflect a strong emphasis on domain credibility and structural properties. *domain_age* and *domain_registration_length* assess the trustworthiness of a domain based on its lifespan and registration period, common indicators of legitimacy. *nb_hyperlinks* again captures page complexity and potential malicious behavior through excessive linking. *page_rank* continues to serve as a proxy for site authority, while *length_hostname* helps detect obfuscated or deceptive domain structures. Together, these features allow LightGBM to effectively capture both reputational and syntactic patterns of malicious URLs.

However, feature importance is usually model-specific and depends on the internal mechanics of a particular algorithm, for example, Gini importance (for RF) and gain-based or split-based importance (for XGBoost and LightGBM). These methods can sometimes be biased, particularly toward features with more categories or higher variance. Our SHAP analysis helps validate the top features identified by traditional feature importance methods. By comparing the top features from both approaches, we can confirm whether they align. If they do, it reinforces the reliability of these features. If they do not, SHAP can help explore whether certain features may have a hidden or nonlinear impact that traditional methods overlook.

Computing the SHAP values for each feature allows us to quantify the contribution of individual features to the models' predictions. These values allow us to gain insights into which features are driving the decision-making process of the classifiers. For instance, a high SHAP value for a specific feature indicates its significant impact on the prediction, while a negative value suggests a counteractive influence.

4.2.1 | RF SHAP Analysis

The SHAP visualizations in Figure 21 and Figure 22 show the beeswarm plots for RF for all the data and the waterfall plot for a single data instance, respectively.

According to the SHAP beeswarm plot, the top five features identified through traditional feature importance—*google_index*, *page_rank*, *nb_hyperlinks*, *web_traffic*, and *nb_www*—demonstrate clear and interpretable influence on the model's output. For *google_index*, lower values (i.e., URLs not indexed by Google) are associated with negative SHAP values, indicated by blue points extending to the left, thus pushing predictions toward the phishing class. Conversely, higher *google_index* values produce positive SHAP values (red points on the right), leading to benign classifications. The impact is asymmetric, with high *google_index* values exerting a stronger positive influence than the negative effect of low values.

A similar pattern is observed for *page_rank*: lower-ranked pages have higher positive SHAP values, increasing the likelihood of phishing classification, while higher-ranked pages contribute less strongly toward benign predictions. In the case of *nb_hyperlinks*, lower values are linked to positive SHAP contributions (favoring benign classification), while higher counts result in negative SHAP values, suggesting that an excessive number of hyperlinks may be indicative of phishing.

For *web_traffic*, only high values form a dense cluster with negative SHAP contributions, whereas low values show more positive SHAP values and a greater influence on phishing predictions—highlighting the role of web popularity in model decisions. Lastly, a higher *nb_www* count tends to produce negative SHAP values, indicating that URLs with more "www" tokens are more likely to be classified as phishing, while lower values are associated with benign predictions.

Overall, the SHAP analysis not only confirms the importance of these features but also provides detailed insights into the direction and magnitude of their influence. Notably, *page_rank* and *web_traffic* emerge as the most impactful features in driving phishing predictions.

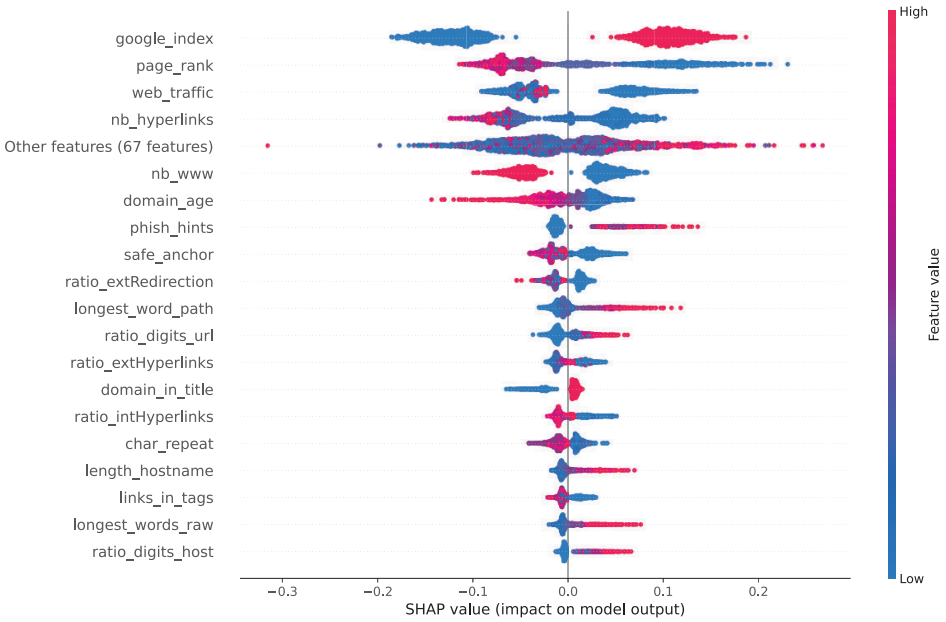

FIGURE 21 | RF beeswarm plot.

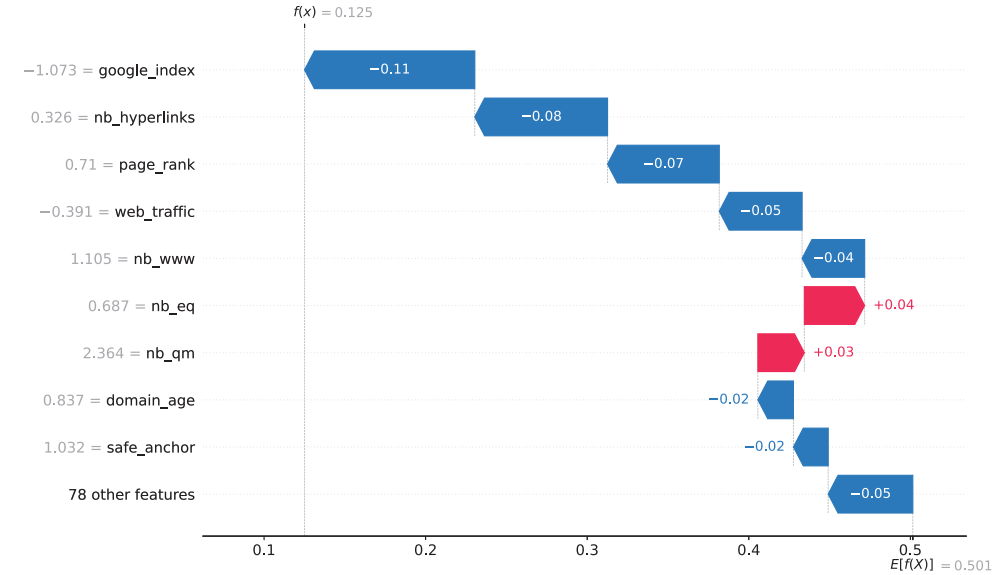

FIGURE 22 | RF waterfall plot.

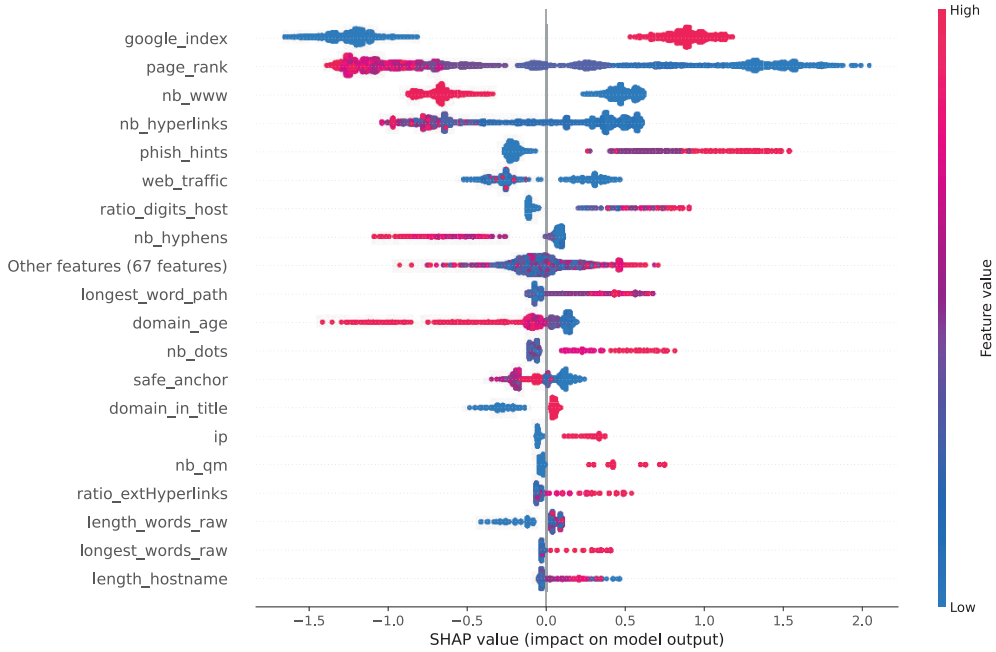

FIGURE 23 | XGBoost beeswarm plot.

The SHAP waterfall plot illustrates how individual features influence a single prediction made by the RF model (local interpretability). Beginning from the model's expected output (base value) of 0.501, the cumulative contributions of specific features lower the final prediction to 0.125—indicating a high confidence in the phishing class. Features such as *google_index*, *nb_hyperlinks*, *page_rank*, and *web_traffic* contribute negatively, each pushing the prediction toward phishing by indicating low reputation or suspicious behavior. Conversely, *nb_qm* (number of question marks) and *nb_eq* (number of equal signs) exert upward influence, increasing the score slightly due to their association with typical phishing URL patterns. However, their positive effect is insufficient to counteract the strong negative contributions from reputation-based and structural features. This local explanation provides transparency into how these features interact to shape the final model output.

The features consistently influential in both global (beeswarm) and local (waterfall) SHAP interpretations are *google_index*, *page_rank*, *nb_hyperlinks*, and *web_traffic*, highlighting their central role in driving the model's phishing predictions.

4.2.2 | XGBoost SHAP Analysis

For XGBoost, Figures 23 and 24 show the beeswarm plots for all the data and the waterfall plot for a single data instance, respectively. The SHAP beeswarm and waterfall plots for the

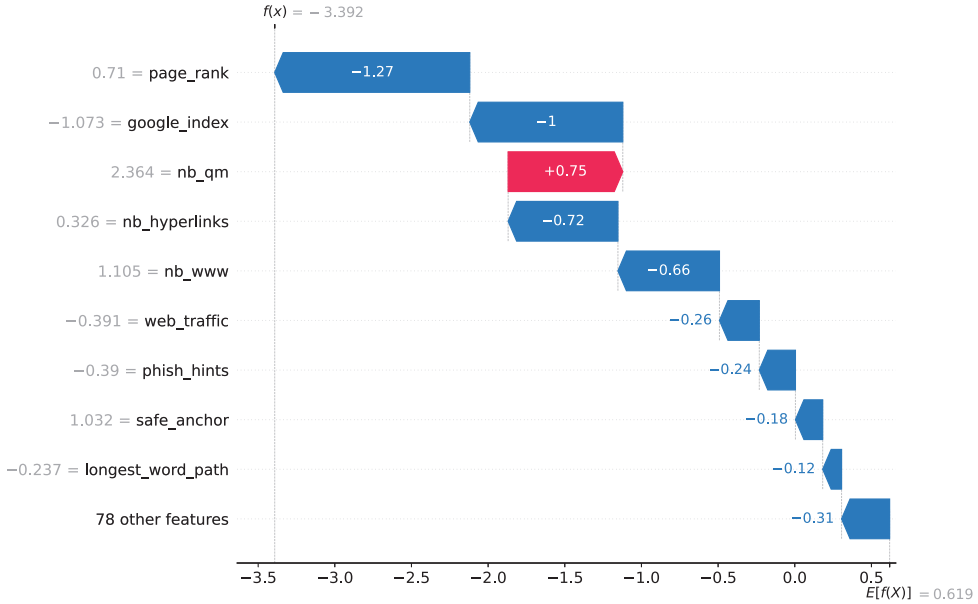

FIGURE 24 | XGBoost waterfall plot.

XGBoost model reveal how the top-ranked features influence both global and local predictions. The top five features by traditional importance—*google_index*, *nb_hyperlinks*, *nb_www*, *page_rank*, and *web_traffic*—show distinct behavior in the SHAP beeswarm plot. For *google_index*, higher values (red points) are associated with positive SHAP values, pushing predictions toward the benign class, while lower values increase phishing likelihood. *Page_rank* follows a similar pattern: lower values (blue) yield strong positive SHAP values, suggesting they increase the chance of phishing classification, while higher values push toward benign. In contrast, *web_traffic* displays an unexpected pattern—higher values (red) are mostly associated with negative SHAP values, implying that even popular sites may be classified as phishing in this dataset. For *nb_hyperlinks* and *nb_www*, higher values tend to have negative SHAP values, consistent with common phishing URL patterns involving excessive links or suspicious domain structures.

In the SHAP waterfall plot for a specific instance, the model prediction shifts dramatically from the base value of 0.169 to a final log-odds score of −3.392, indicating a highly confident phishing classification. This drop is driven by strong negative contributions from several top features: *page_rank* (−1.27), *google_index* (−1.00), *nb_hyperlinks* (−0.72), *nb_www* (−0.66), and *web_traffic* (−0.26), among others like *phish_hints*, *safe_anchor*, and *longest_word_path*. The only feature with a positive SHAP value is *nb_qm* (+0.75), which slightly offsets the negative shift but is insufficient to reverse the overall direction. This local explanation confirms that the model's phishing prediction is driven by a combination of poor reputation scores and suspicious structural patterns, aligning closely with the global SHAP analysis.

4.2.3 | LightGBM SHAP Analysis

For XGBoost, Figures 25 and 26 show the beeswarm plots for all the data and the waterfall plot for a single data instance, respectively. The SHAP beeswarm and waterfall plots for the LightGBM model reveal the most influential features in both global and local prediction contexts. According to the beeswarm plot, the top contributors to the model's predictions include *google_index*, *page_rank*, *nb_www*, *nb_hyperlinks*, and *web_traffic*. In the beeswarm plot, higher values of *google_index* (red points) are associated with positive SHAP values, contributing to benign predictions, while lower values increase phishing risk. A reverse trend is observed for *page_rank*, where lower values (blue points) push the prediction toward phishing, while higher values help indicate benign sites. *web_traffic* also shows that low-traffic domains lean toward phishing predictions. Structural features like *nb_hyperlinks* and *nb_www* contribute negatively when their values are high, consistent with known phishing characteristics.

The SHAP waterfall plot illustrates how individual features contribute to a specific prediction made by the LightGBM model. Starting from a base value of 0.275 (the model's expected output in log-odds across the dataset), the prediction sharply decreases to −9.023, indicating a highly confident classification of the instance as phishing. This decline is driven by a series of strong negative SHAP contributions. The largest impact comes from *domain_age* (−2.49), followed by *page_rank* (−1.92) and *google_index* (−1.46), suggesting that the domain is newly created, poorly ranked, and not indexed—characteristics strongly associated with phishing websites. Additional negative contributions from features such as *nb_hyperlinks* (−0.66), *domain_registration_length* (−0.34), and *length_hostname* (−0.15) further reinforce this classification. In contrast, features like *shortest_word_path* (+0.29) and *nb_hyphens* (+0.14) slightly pull the prediction toward benign, but their impact is too small to counteract the dominant phishing indicators. Overall, the plot demonstrates how LightGBM relies heavily on domain credibility and structural URL signals to reach a confident phishing decision in this instance.

The comparison of the SHAP result across all models, based on the top 10 globally influential features, reveals both consistent

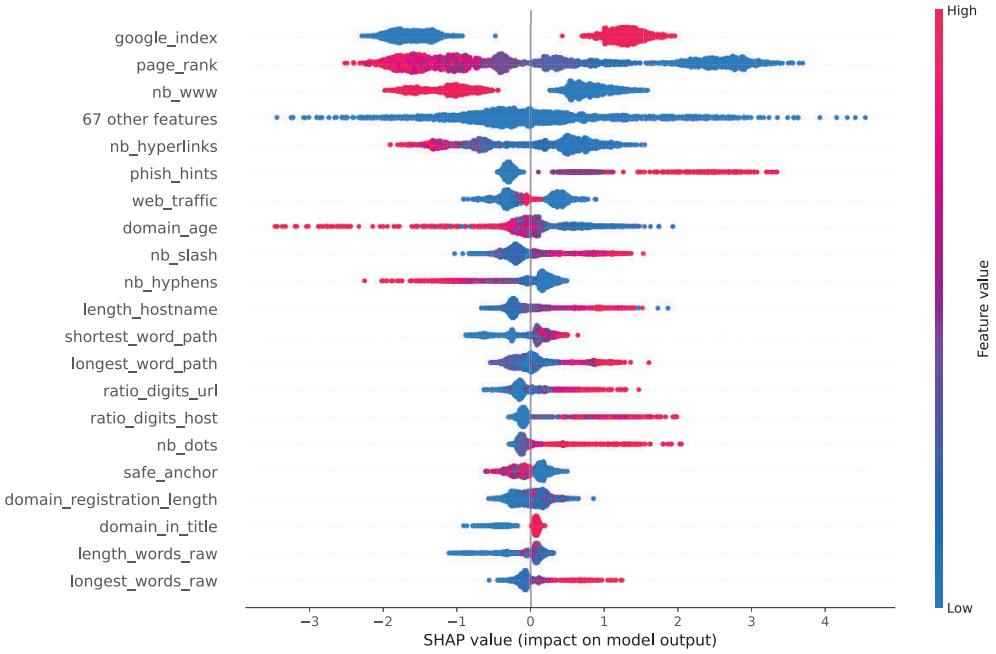

FIGURE 25 | LightGBM beeswarm plot.

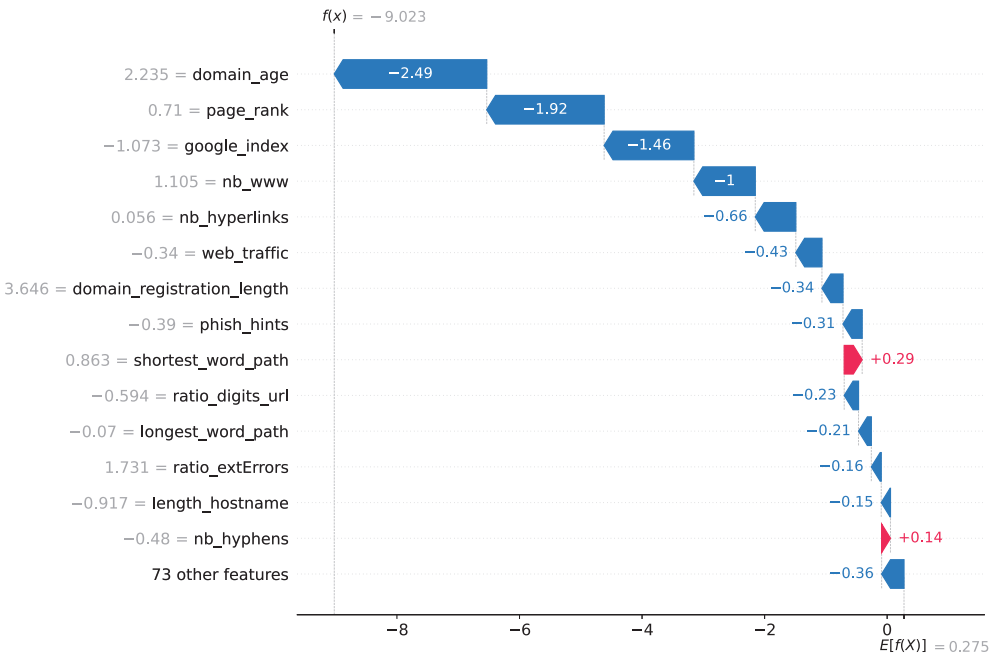

FIGURE 26 | LightGBM waterfall plot.

and distinctive patterns in global feature importance for phishing URL detection. Notably, *google_index* and *page_rank* appear as the top two most influential features across all three models, indicating strong agreement that web credibility and indexing status are highly predictive indicators of phishing activity. Furthermore, features such as *nb_www*, *nb_hyperlinks*, and *phish_hints* are consistently ranked among the top contributors in all models, suggesting that structural patterns and phishing-specific content remain critical cues.

In terms of model-specific emphasis, the RF model gives relatively higher weight to *web_traffic* and *safe_anchor*, potentially reflecting its sensitivity to user engagement and link behavior. XGBoost, on the other hand, uniquely highlights *ratio_digits_host*, *nb_hyphens*, and *longest_word_path*, pointing to a greater focus on character-level anomalies within URLs. LightGBM shares much of its top-ranked feature set with XGBoost but also prioritizes *nb_slash* and brings back *domain_age*, suggesting a nuanced balance between lexical patterns and domain metadata.

Overall, while there is strong consistency in the core influential features across all models, the variation in secondary features reflects each model's inherent learning bias and capacity to detect

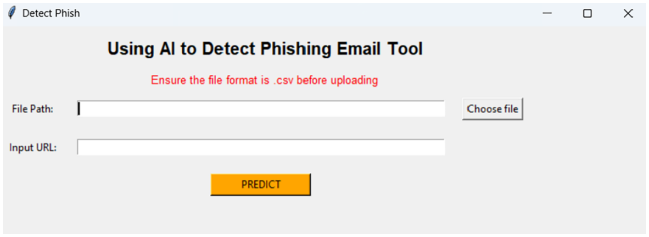

FIGURE 27 | Desktop interface.

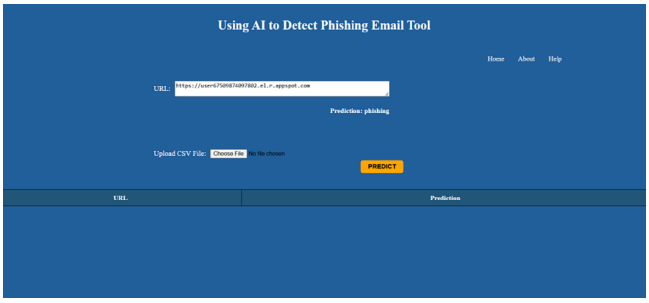

FIGURE 28 | Web interface.

different types of phishing patterns. This diversity may offer complementary strengths in ensemble or hybrid approaches.

4.3 | Flask Application Interface

The phishing detection tool is designed for both desktop (Figure 27) and web (Figure 28) platforms. The web version does not require installation. Users can either enter a single URL directly in the browser or upload a CSV file for batch detection. The system then predicts phishing attempts using a machine learning model running on a Flask backend. Upon clicking the *predict* button, the system evaluates whether the entered URL is phishing or legitimate. Using Flask has its advantages as well as its limitations. Due to its lightweight core framework, building complex systems often requires relying on a large number of third-party libraries, which may increase code maintenance costs. Also, Flask does not natively support asynchronous processing, which can impact performance when handling high-concurrency requests. Over the long term, the maintenance cost may also be relatively high. Flask is used here to quickly verify the effectiveness of the application and visually demonstrate the use of the model. If the system needs to be put into mass production, further optimization of the architecture will be necessary, and more robust and scalable technology stacks should be adopted, such as Nginx for load balancing, Redis to cache API results, FastAPI for asynchronous task processing, and Kubernetes for automatic scaling. These measures will ensure that the system can efficiently and stably handle large-scale traffic (Figure 28).

5 | Conclusion

Implementing machine learning techniques for phishing email detection using URLs demonstrates significant promise in enhancing cybersecurity. This paper develops and evaluates three machine learning models, RF, XGBoost, and LightGBM, to predict phishing emails from URL information. In addition to evaluating our models performance across multiple datasets to assess generalizability, this study incorporates SHAP-based explainability to interpret model behavior at both global and local levels. Among the models tested, RF demonstrates the best generalization overall, achieving the lowest average misclassification rate (≈2.90%) across all test sets, followed by XGBoost (≈4.57%) and LightGBM (≈5.42%). Despite these differences, all our models achieve an average detection accuracy of approximately 96%, indicating strong learning and generalization from the engineered features. The SHAP analysis further reinforced these results by identifying key features–such as *google_index* and *page_rank*–as consistently influential across models and datasets. This alignment between performance metrics and interpretability insights strengthens confidence in the robustness of the models and supports their suitability for deployment in real-world, security-critical phishing detection scenarios. Furthermore, we develop a tool using the Flask framework to build back-end services that handle requests from desktop and web clients. The back-end service integrates the RF and XGBoost models via Flask to support complex data processing needs. The Flask server provides two main API endpoints: one for real-time predictions of individual data points and another for batch predictions of uploaded CSV files.

Funding

The authors have nothing to report.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. R. B. Basnet and A. H. Sung, *Classifying Phishing Emails Using Confidence-Weighted Linear Classifiers, in: International Conference on Information Security and Artificial Intelligence (ISAI)* (Citeseer, 2010), 108–112.

2. A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A Comprehensive Survey of AI-Enabled Phishing Attacks Detection Techniques," *Telecommunication Systems* 76 (2021): 139–154.

3. T. Gangavarapu, C. Jaidhar, and B. Chanduka, "Applicability of Machine Learning in Spam and Phishing Email Filtering: Review and Approaches," *Artificial Intelligence Review* 53, no. 7 (2020): 5019–5081.

4. A. Apwg, *Phishing Activity Trends Report: 1st Quarter 2024* (Anti-Phishing Working Group, 2021).

5. G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, *On the Effectiveness of Machine and Deep Learning for Cyber Security, in: 2018 10th International Conference on Cyber Conflict (CyCon)* (IEEE, 2018), 371–390.

6. S. Durgaraju, D. V. T. Vel, and H. Madathala, "The Evolution of Cyber Threats and Defenses: A Review of Innovations and Challenges," in *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)* (IEEE, 2025), 117–123.

7. N. R. Choudhury, S. Paul, and S. Ghosh, "Comparative Analysis of Traditional vs. AI-Driven Network Security," in *AI for Large Scale Communication Networks* (IGI Global, 2025), 107–128.

8. A. Al Siam, M. M. Hassan, and T. Bhuiyan, *Artificial Intelligence for Cybersecurity: A State of the Art, in: 2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)* (IEEE, 2025), 1–7.

9. H. Dong and I. Kotenko, "Cybersecurity in the AI Era: Analyzing the Impact of Machine Learning on Intrusion Detection," *Knowledge and Information Systems* 67 (2025): 3915–3966.

10. T. Hornyak, "Strength From Weakness," *Nature* 555, no. 7697 (2018): S61.

11. M. Ozkan-Okay, E. Akin, Ö. Aslan, et al., "A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions," *IEEE Access* 12 (2024): 12229–12256.

12. B. Xua and G. Yang, "Interpretability Research of Deep Learning: A Literature Survey," *Information Fusion* 115 (2024): 102721.

13. M. T. Hosain, J. R. Jim, M. Mridha, and M. M. Kabir, "Explainable AI Approaches in Deep Learning: Advancements, Applications and Challenges," *Computers and Electrical Engineering* 117 (2024): 109246.

14. T. Bolukbasi, A. Pearce, A. Yuan, et al., An Interpretability Illusion for Bert, arXiv preprint arXiv:2104.07143 (2021).

15. L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?," *Advances in Neural Information Processing Systems* 35 (2022): 507–520.

16. R. Shwartz-Ziv and A. Armon, "Tabular Data: Deep Learning Is Not All You Need," *Information Fusion* 81 (2022): 84–90.

17. B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do Imagenet Classifiers Generalize to Imagenet?," in *International Conference on Machine Learning* (PMLR, 2019), 5389–5400.

18. Y. Zhang, Q. V. Liao, and R. K. Bellamy, "Effect of Confidence and Explanation on Accuracy and Trust Calibration in Ai-Assisted Decision Making," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (ACM, 2020), 295–305.

19. J. Ohse, B. Hadžić, P. Mohammed, et al., "Zero-Shot Strike: Testing the Generalisation Capabilities of Out-Of-The-Box Llm Models for Depression Detection," *Computer Speech & Language* 88 (2024): 101663.

20. A. Sundararaj and G. Kul, "Impact Analysis of Training Data Characteristics for Phishing Email Classification," *Journal of Wireless Mobile Networks Ubiquitous Computing and Dependable Applications* 12, no. 2 (2021): 85–98.

21. J. Zhan and L. Thomas, "Phishing Detection Using Stochastic Learning-Based Weak Estimators," in *2011 IEEE Symposium on Computational Intelligence in Cyber Security (CICS)* (IEEE, 2011), 55–59.

22. R. S. Rao and A. R. Pais, "Detection of Phishing Websites Using an Efficient Feature-Based Machine Learning Framework," *Neural Computing and Applications* 31 (2019): 3851–3873.

23. O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine Learning Based Phishing Detection From Urls," *Expert Systems With Applications* 117 (2019): 345–357.

24. M. Dewis and T. Viana, "Phish Responder: A Hybrid Machine Learning Approach to Detect Phishing and Spam Emails," *Applied System Innovation* 5, no. 4 (2022): 73.

25. Y.-H. Chen and J.-L. Chen, "AI@ Ntiphish—Machine Learning Mechanisms for Cyber-Phishing Attack," *IEICE Transactions on Information and Systems* 102, no. 5 (2019): 878–887.

26. I. Fette, N. Sadeh, and A. Tomasic, "Learning to Detect Phishing Emails," in *Proceedings of the 16th International Conference on World Wide Web* (Association for Computing Machinery, 2007), 649–656.

27. Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism," *IEEE Access* 7 (2019): 56329–56340.

28. E. S. Gualberto, R. T. De Sousa, T. P. D. B. Vieira, J. P. C. L. Da Costa, and C. G. Duque, "The Answer Is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering," *IEEE Access* 8 (2020): 223529–223547.

29. A. A. Akinyelu and A. O. Adewumi, "Classification of Phishing Email Using Random Forest Machine Learning Technique," *Journal of Applied Mathematics* 2014, no. 1 (2014): 425731.

30. S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A Comparison of Machine Learning Techniques for Phishing Detection," in *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit* (Association for Computing Machinery, 2007), 60–69.

31. B. Hughes, S. Bird, H. Lee, and E. Klein, "Experiments With Data-Intensive NLP on a Computational Grid," in *Proceedings of the International Workshop on Human Language Technology* (Citeseer, 2004), http://eprints.unimelb.edu.au/archive/00000503.

32. C. Coyotes, V. S. Mohan, J. Naveen, R. Vinayakumar, K. Soman, and A. Verma, "ARES: Automatic Rogue Email Spotter," in *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal. (IWSPA)* (Association for Computing Machinery, 2018).

33. M. Hiransha, N. A. Unnithan, R. Vinayakumar, K. Soman, and A. Verma, "Deep Learning Based Phishing e-Mail Detection," in *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal. (IWSPA)* (Association for Computing Machinery, 2018), 1–5.

34. A. K. Jain and B. B. Gupta, "Towards Detection of Phishing Websites on Client-Side Using Machine Learning Based Approach," *Telecommunication Systems* 68 (2018): 687–700.

35. S. Bagui, D. Nandi, S. Bagui, and R. J. White, "Machine Learning and Deep Learning for Phishing Email Classification Using One-Hot Encoding," *Journal of Computer Science* 17, no. 7 (2021): 610–623.

36. L. Shalini, S. S. Manvi, N. C. Gowda, and K. Manasa, "Detection of Phishing Emails Using Machine Learning and Deep Learning," in *2022 7th International Conference on Communication and Electronics Systems (ICCES)* (IEEE, 2022), 1237–1243.

37. N. A. Unnithan, N. Harikrishnan, R. Vinayakumar, K. Soman, and S. Sundarakrishna, "Detecting Phishing e-Mail Using Machine Learning Techniques," in *Proc. 1st Anti-Phishing Shared Task Pilot 4th Acm Iwspa Co-Located 8th ACM Conf. Data Appl. Secur. Privacy (Codaspy)* (Association for Computing Machinery, 2018), 51–54.

38. P. Bhatti, Z. Jalil, and A. Majeed, "Email Classification Using Lstm: A Deep Learning Technique," in *2021 International Conference on Cyber Warfare and Security (ICCWS)* (IEEE, 2021), 100–105.

39. V. Ra, B. G. HBa, A. K. Ma, S. K. KPa, P. Poornachandran, and A. Verma, "Deepanti-Phishnet: Applying Deep Neural Networks for Phishing Email Detection," in *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)* (Association for Computing Machinery, 2018), 1–11.

40. X. Wang, W. Zhang, and S. Rajtmajer, Monolingual and Multilingual Misinformation Detection for Low-Resource Languages: A Comprehensive Survey, arXiv Preprint arXiv:2410.18390 (2024).

41. M. Somesha and A. R. Pais, "Classification of Phishing Email Using Word Embedding and Machine Learning Techniques," *Journal of Cyber Security and Mobility* 11, no. 3 (2022): 279–320.

42. S. Atawneh and H. Aljehani, "Phishing Email Detection Model Using Deep Learning," *Electronics* 12, no. 20 (2023): 4261.

43. N. Altwaijry, I. Al-Turaiki, R. Alotaibi, and F. Alakeel, "Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models," *Sensors* 24, no. 7 (2024): 2077.

44. A. Yasin and A. Abuhasan, "An Intelligent Classification Model for Phishing Email Detection," *International Journal of Network Security & Its Applications* 8 (2016): 1–18.

45. U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, "Cloud-Based Email Phishing Attack Using Machine and Deep Learning Algorithm," *Complex & Intelligent Systems* 9, no. 3 (2023): 3043–3070.

46. R. Alotaibi, I. Al-Turaiki, and F. Alakeel, "Mitigating Email Phishing Attacks Using Convolutional Neural Networks," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)* (IEEE, 2020), 1–6.

47. S. S. Shafin, "An Explainable Feature Selection Framework for Web Phishing Detection With Machine Learning," *Data Science and Management* 8, no. 2 (2024): 127–136.

48. P. R. G. Hernandes, C. P. Floret, K. F. C. De Almeida, V. C. Da Silva, J. P. Papa, and K. A. P. Da Costa, "Phishing Detection Using Url-Based Xai Techniques," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2021), 1–6.

49. N. Puri, P. Saggar, A. Kaur, and P. Garg, "Application of Ensemble Machine Learning Models for Phishing Detection on Web Networks," in *2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)* (IEEE, 2022), 296–303.

50. F. Greco, G. Desolda, and A. Esposito, Explaining Phishing Attacks: An XAI Approach to Enhance User Awareness and Trust, in: ITASEC (2023).

51. A. M. Salih, Z. Raisi-Estabragh, I. B. Galazzo, et al., "A Perspective on Explainable Artificial Intelligence Methods: Shap and Lime," *Advanced Intelligent Systems* 7 (2024): 2400304.

52. T. Tiwari, Phishing Site URLs: Malicious and Phishing Attacks URLs (2020), https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls/data.

53. Winson, Dataset for Link Phishing Detection (2024), https://www.kaggle.com/datasets/winson13/dataset-for-link-phishing-detection/data.

54. K. Manish, Web Page Phishing Detection (2020), https://www.kaggle.com/datasets/manishkc06/web-page-phishing-detection.

55. M. Mia, D. Derakhshan, and M. M. A. Pritom, "Can Features for Phishing URL Detection Be Trusted Across Diverse Datasets? A Case Study With Explainable AI," in *Proceedings of the 11th International Conference on Networking, Systems, and Security* (Association for Computing Machinery, 2024), 137–145.

56. M. Nanda, M. Saraswat, and P. K. Sharma, "Enhancing Cybersecurity: A Review and Comparative Analysis of Convolutional Neural Network Approaches for Detecting Url-Based Phishing Attacks," *E-Prime-Advances in Electrical Engineering, Electronics and Energy* 8 (2024): 100533.

57. M. Nanda and S. Goel, "Url Based Phishing Attack Detection Using Bilstm-Gated Highway Attention Block Convolutional Neural Network," *Multimedia Tools and Applications* 83, no. 27 (2024): 69345–69375.

58. D. Gómez and A. Rojas, "An Empirical Overview of the No Free Lunch Theorem and Its Effect on Real-World Machine Learning Classification," *Neural Computation* 28, no. 1 (2016): 216–228.