The Journal of Engineering

ORIGINAL RESEARCH OPEN ACCESS

Explainable Machine Learning for Phishing Site Detection: A High-Efficiency Approach Using Boosting Models and SHAP

Khandaker Mohammad Mohi Uddin[1] | Nitish Biswas[2] | Sarreha Tasmin Rikta[2] | Md. Nur -A-Alam[3] | Rafid Mostafiz[4]

[1]Department of Computer Science and Engineering, Southeast University, Dhaka, Bangladesh | [2]Department of Computer Science and Engineering, Dhaka International University, Dhaka, Bangladesh | [3]Department of Computer Science and Engineering, Computer Science and Engineering, Sunamganj Science and Technology University, Sylhet, Bangladesh | [4]Institute of Information Technology, Noakhali Science and Technology University, Noakhali, Bangladesh

Correspondence: Khandaker Mohammad Mohi Uddin (jilanicsejnu@gmail.com)

Received: 9 April 2025 | **Revised:** 17 June 2025 | **Accepted:** 5 July 2025

Funding: The authors received no specific funding for this work.

ABSTRACT

Nowadays, a wide range of electronic devices are being connected through the internet, and a wide range of cybercrimes are being coordinated. Phishing is one of the most serious online crimes. It poses a major threat to people and businesses, resulting in enormous financial losses and user-provided data breaches. Traditional phishing detection methods are time-consuming and often fail to detect novel phishing approaches. This study aims to develop an efficient and explainable machine learning model to detect phishing websites, providing transparency in predictions to increase user trust. To tackle this challenge, we suggest a highly efficient phishing detection method that uses ensemble learning techniques, specifically gradient boosting machine, extreme gradient boosting, and light gradient boosting, combined with Shapley additive explanations for clarity. The innovation comes from integrating SHAP with ensemble models to provide high accuracy and clear explanations, which helps end-users and security experts comprehend the decision-making process. Using a publicly available phishing dataset from Kaggle, the XGBoost model achieved a high accuracy of 97.78%. It outperformed other models. The SHAP-based analysis visually explained the features of the importance and contribution of each classifier. This approach promotes model transparency and improves the reliability of the phishing detection system.

1 | Introduction

Life has become easier as a consequence of the advancement of digital communication and the move toward digitalization. Government and non-government organizations, companies, and industries conduct their entire activity online. The internet has emerged as the main factor behind this era. Cybercrime is growing concurrently with that situation, day by day. As a result, it causes financial losses, privacy intrusions, identity theft, and the disruption of crucial infrastructure. Cybercrime is a serious threat to people, corporations, and governments. Phishing and scams, identity theft, ransomware attacks, hacking, and inappropriate use of computer networks, and online fraud are all common types of cybercrimes. However, the exclusive subject of this study is phishing scams. According to cybersecurity experts, phishing is a risk that can result in data theft, credit card fraud, malware infections, and other significant monetary losses for consumers as well as companies [1–3]. Phishing can take place via email, SMS, URLs, phone calls, or websites, among other channels. These techniques are employed to deceive

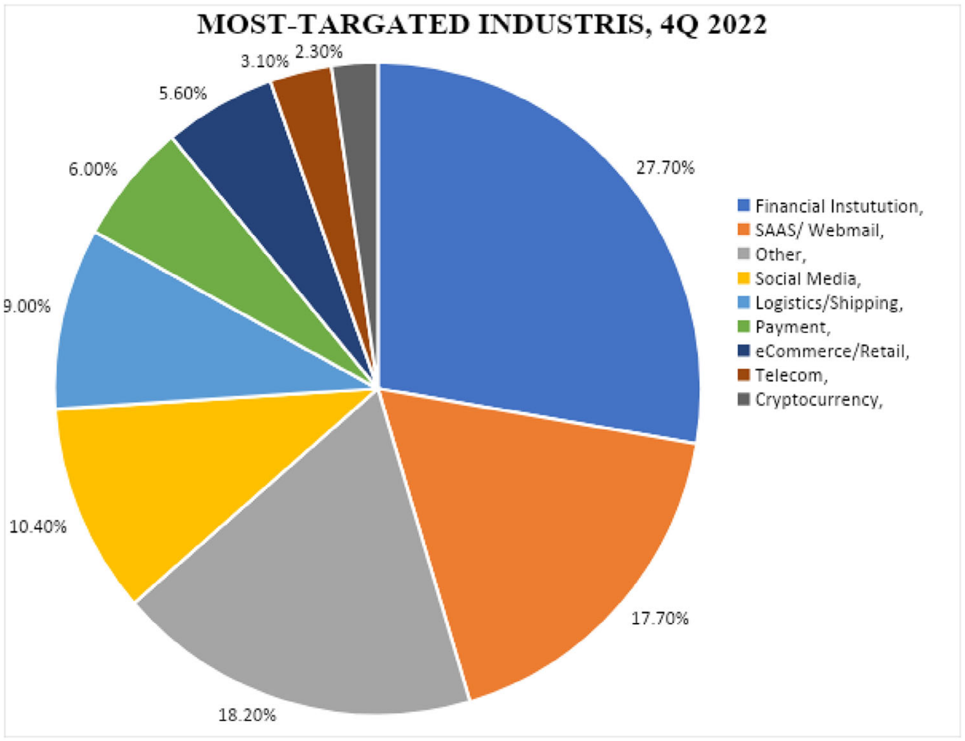

FIGURE 1 | Most targeted industries in the fourth quarter-2022.

users into installing malicious software or disclosing private information, such as financial accounts, credit card numbers, or login information.

Anti-phishing working group (APWG) reported that the largest phishing attack record year was 2022, with 4.7 million attacks [4]. This growth has exceeded 150% annually since the beginning of 2019. They recorded a total of 1,350,037 phishing assaults in the last 3 months of 2022. APWG discovered that the financial sector had the most scams in the final quarter of 2022, accounting for 27.7% of the total phishing, compared to 23.2 percent in Q32022. Then, with 17.7% of all attacks—a tiny decrease from Q3—webmail was the industry most frequently hit by phishing attacks. Social media is the fourth-highest phishing-affected area, while shipping and logistics are the fifth-highest impacted areas, respectively. Figure 1 displays the industry's most frequently targeted segments in the fourth quarter of 2022.

According to the illegal activity report for 2022 from the FBI (Federal Bureau of Investigation), phishing was the most common type of cyberattack from 2019 to 2022 [5]. The FBI's Internet Crime Complaint Center (IC3) recorded 800,944 allegations in 2022, a 5% decrease from 2021. However, from $6.9 billion in 2021 to $10.3 billion in 2022, the possible total loss has grown by $3.4 billion. These statistics show how dangerous phishing attempts have evolved over time. Therefore, more effort is needed to find these problems. Numerous organizations rely on human skills to spot these threats. But the problem is that it may be quite difficult for human eyes to discern, even for experts. Cybersecurity experts usually focus on message attachments to spot phishing attacks. Furthermore, it takes too long to find. These characteristics have inspired many scholars to investigate this topic and have demonstrated a variety of effective techniques for addressing both established and new fraud. However, the majority of studies suggested categorizing and recognizing possible phishing attacks using patterns and attributes in order to identify phishing attempts employing machine learning methods.

Despite many studies, machine learning-based phishing detection still faces significant limitations. Many models function as "black boxes," lacking accountability, transparency, and interpretability [6, 7], which impedes cybersecurity experts' ability to trust and comprehend one another. Most current systems focus on accuracy but often overlook the need for explainability. This gap makes practical adoption difficult. Few methods provide insights into how the models make decisions or show underlying patterns and trends.

While significant progress has been made in using machine learning for phishing detection, current models mainly focus on accuracy, which sacrifices interpretability, fairness, and trust. This limitation reduces their usefulness in real-world cybersecurity situations. Recent studies have started to include explainable AI techniques like SHAP [8] and LIME. However, these efforts often focus on narrow goals, such as analysing feature transferability [9] across datasets or improving feature selection [10] without establishing a practical framework for deployment. Furthermore, many methods lack instance-level interpretability, rely on default model settings, and do not use tuning techniques like grid search and cross-validation. These techniques are essential for developing robust and flexible systems.

Our study tackles these issues by introducing a phishing detection framework based on XGBoost. This framework offers both global and local SHAP-based explanations alongside strong predictive performance. To ensure transparency, reliability, and real-time

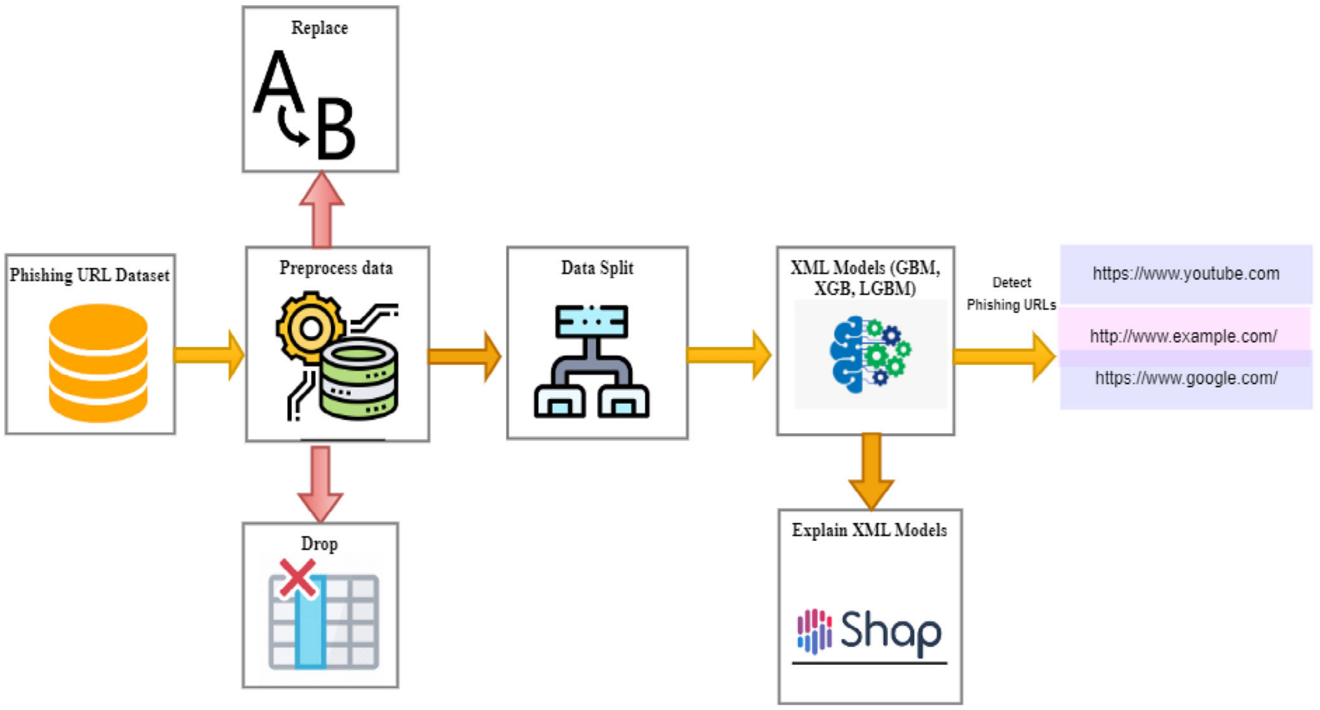

FIGURE 2 | Proposed approach for detecting phishing attacks.

application, it is supported by thorough hyperparameter optimization. Based on these motivations, the study is guided by the three major research questions:

- How can a machine learning model like XGBoost be used to detect phishing URLs based on web-based features?
- How can SHAP help explain the predictions of phishing detection models and identify the most important features?
- To what extent does adding explainability to phishing detection models improve trust and usability for cybersecurity professionals?

Figure 2 displays the proposed technique for identifying phishing attempts.

The overall reflection of XAI phishing is shown in this figure. The phishing URL dataset, which goes through preprocessing, had to first be loaded. At this stage, the output of the dataset is replaced, and extra columns are eliminated. After data preparation, the data was split into two parts. Finally, phishing URLs were identified using machine learning classifiers, and SHAP was utilized to explain the behaviour of those models. This article makes the following contributions to this study:

- A fully explainable phishing detection framework is presented that combines SHAP with three ensemble classifiers, GBM, XGBoost, and LightGBM. This approach improves transparency and trust in predictions.
- Unlike previous work that focuses only on accuracy, our method balances predictive performance with interpretability. This balance is crucial for real-world cybersecurity applications.
- SHAP visualizations, including global feature importance, beeswarm, and local waterfall plots are used to give both global and instance-level insights into how the model behaves.
- The XGBoost plus SHAP model achieves top accuracy of up to 97.78% on a balanced Kaggle phishing dataset, outperforming existing methods in accuracy, precision, recall, and F1-score.
- Grid search and 10-fold cross-validation are employed for hyperparameter tuning, ensuring model robustness and generalizability.
- By integrating explainability into high-performance models, this framework supports ethical AI and promotes transparent, accountable decision-making in cybersecurity.

The remaining sections of this study are divided into five sections. Section 2 shows the related work, and Section 3 covers materials and methods such as explainability, SHAP, hyperparameter tuning, utilized machine-learning models, evaluation metrics, and data set specification and pre-processing. Sections 4 and 5 deal with the results analysis and discussion, respectively. This task is finally completed in Section 6.

2 | Related Works

The largest online hazard nowadays is phishing. Several researchers have attempted to offer a variety of facilities to protect consumers from cyber-attacks through the utilization of machine learning, blacklists, deep learning, and whitelists. Despite having so many options, there are still limitations due to a lack of accountability, trustworthiness, and fairness. There hasn't been a lot of study done on explainable machine learning in phishing scam findings. However, some current research on detecting phishing attacks is highlighted here.

Mia et al. [9] investigated the reliability of phishing detection features across multiple datasets using SHAP for explainability. By examining two public datasets and using explainable AI tools like SHAP, the authors demonstrate that many features rely on the dataset. Models trained on one dataset often do not work well on another, even if they have similar features. The study points out the limited generalizability of features and emphasizes the need for caution when using phishing detection models with various data sources.

Calzarossa et al. [10] proposed a clear machine learning method for detecting phishing websites. They used a random forest classifier, along with a new feature selection method based on Lorenz Zonoids, a multidimensional version of the Gini coefficient. Their study focused on finding the most important URL-based features for phishing detection through exploratory analysis and statistical interpretation. That model performed well with a smaller set of features; however, its explainability was limited to overall feature importance, as measured by Gini and Lorenz metrics. It did not provide insights for individual instances or SHAP-based visualization.

Jovanovic et al. [11] introduced a hybrid two-level framework for detecting phishing websites using an improved method called the diversity-oriented firefly algorithm (DOFA). The framework carries out feature selection and XGBoost hyperparameter tuning in two steps. It was tested on three datasets—two from Mendeley and one from UCI—and achieved high accuracies of 94.46%, 95.54%, and 97.60%. The proposed method outperformed several leading algorithms and applied SHAP for explainability, pinpointing key features like SSL certificates and anchor URLs. This approach effectively balances performance and clarity in phishing detection.

In another study [12], the author presents a new explainable feature selection (FS) framework named SLA-FS. It combines SHAP, which offers global interpretability, and LIME, which provides local insights, to find and select the most useful features for detecting phishing websites. The framework is tested with random forest (RF), XGBoost (XGB), and K-nearest neighbours (KNN) on a recent web phishing dataset from Mendeley. SLA-FS shows better classification accuracy and shorter detection time than traditional FS methods. The highest accuracy reached is 97.41% with RF after feature reduction.

Babic et al. [13] suggested a click fraud detection framework that uses recurrent neural networks (RNNs) optimized by an adapted crayfish optimization algorithm (ACOA). The model was tested on a balanced, downsampled Kaggle clickstream dataset and reached a top accuracy of 78.2%. By fine-tuning RNN hyperparameters, ACOA surpassed several other metaheuristic methods and effectively identified sequential patterns in click behaviour, which improved the detection of fraudulent clicks.

Vyvaswini et al. [14] proposed a machine-learning technique for detecting phishing URLs by extracting and analysing various features of authentic and phishing URLs. They suggested five possible methods to assess the URLs, incorporating gradient boosting, random forest, XGBoost, AdaBoost, and support vector machine. They have demonstrated that the existing method's accuracy is roughly 94% and that the random forest classifier increased that accuracy in this trial to 95%. They contend that using this strategy yields accurate outcomes.

Nazmul et al. [15] established another machine learning technique based on feature analysis for detecting phishing attacks. Two well-known machine learning algorithms—decision trees and random forests—were developed to detect phishing attempts using machine learning approaches. The dataset, comprising 11,054 instances and 32 columns, was obtained from Kaggle to conduct the experiment. The properties of the dataset were examined using principal component analysis (PCA). To assess how well these two algorithms performed, the confusion matrix was finally used. Finally, the random forest algorithm achieved an optimal accuracy of 97% in that experiment.

Basit et al. [16] suggested an ensemble method in a different study to identify phishing attacks on websites using machine learning methods. They employed various machine learning classifiers, such as the artificial neural network, decision tree (C4.5), and K-nearest neighbours, together with the random forest classifier. There are 11,055 instances and 30 characteristics in this collection. Performance was measured by 10-fold cross-validation. The results of the experiments demonstrate that the ensemble of KNN with RFC delivers an accuracy rate of 97.33%.

Jian et al. [17] suggested and tested a system using different learning classifiers like support vector machine, AdaBoost, decision tree, and random forest. The goal of this research was to generate machine-learning algorithms that would allow for computerized site layout-based detection of phishing solutions. They claimed that by using CSS layout attributes, their method automatically trains classifiers to identify web page similarity. The results, however, were assessed using the four criteria: F1 score, accuracy, recall, and precision, and it was discovered that random forest surpasses the other three when all four metrics are included.

A number of machine learning classification models, namely decision tree, multi-layer perceptrons, light BGM, random forest, SVM, CatBoost, and XGBoost, are employed in this work [18]. These classifiers identify phishing attempts and record their benefits and drawbacks. For this experiment, the dataset was obtained by downloading the MillerSmiles and PhishTank libraries. The light BGM, however, surpasses the other classifiers with a greater accuracy of 85.5%.

Mahajan and Siddavatam [19] introduced a machine-learning approach for detecting malicious URLs based on extracting and contrasting several features of legitimate and fraudulent URLs. Different kinds of machine-learning-based classifiers are used here, which include the random forest, support vector machine, and decision tree approaches. This study examines the false negative, accuracy, and false negative rates of multiple machine learning classifiers for detecting phishing URLs and selecting the best one. URLs for phishing websites were discovered on www.phishtank.com. The data collection contains 17,058 safe URLs and 19,653 fake URLs, for an overall count of 36,711 URLs. In this investigation, they were able to achieve 97.14% detection accuracy by employing a random forest strategy with the smallest level of false positives.

In a separate study, Gaoqing et al. [20] propose a technique for transforming phishing emails based on structural and communication interaction. Six resource generators are used in this paper, along with a communication relationship selector. In this experiment, the Enron dataset is utilized. This paper's major objective was to feed better data to the model. Here, classification techniques that use random forests are used. After 10-fold cross-validation, the original dataset's accuracy was 96.72%.

They also proposed a machine learning-based technique to detect phishing attacks in [21]. To halt phishing URLs and protect users, they employed and constructed a number of machines learning algorithms, including the decision tree, random forest, K-neighbours classifier, and suggested a hybrid LSD model (LR+SVC+DT). The canopy feature selection method was applied here. This experiment's dataset was taken from Kaggle. Furthermore, a range of evaluation criteria, including specificity, precision, F1-score, accuracy, and recall, are used to assess performance. The accuracy achieved by the naive Bayes algorithms was the greatest at 88.39%.

The studies described above mostly focus on using machine learning strategies to detect phishing attacks. Most of these studies have no interpretability, trustworthiness, or fairness, which is a significant drawback.

3 | Materials and Methods

Figure 3 depicts the working phases of the suggested model for detecting phishing attacks. This study proposes XAI-based phishing attack detection using three gradient boosting classifiers: XGBoost, LightGBM, and gradient boosting machine (GBM). These models were chosen for their strong performance and good compatibility with SHAP, which allows for effective model interpretation. They are widely used in cybersecurity because they balance speed, scalability, and transparency. Other boosting algorithms, such as CatBoost and AdaBoost, were considered but not used. CatBoost is better for categorical data, while AdaBoost often struggles with imbalanced or noisy datasets.

The primary purpose of this study is to describe the models utilizing SHAP so that people comprehend how the model is run and how its results are obtained. This will boost confidence in the model's forecasts among cyber experts and victims. The whole output of this experiment is shown in Algorithm 1 for easier understanding.

3.1 | Dataset Description and Data Pre-Processing

In this study, we used the Kaggle Phishing Website Dataset [22], which includes 11,054 instances with a balanced mix of phishing and legitimate URLs. We chose this dataset because it has 32 clear, interpretable features, such as having_IP_Address, SSLfinal_State, URL_Length, and Prefix_Suffix. These features come from domain-specific phishing indicators and work well

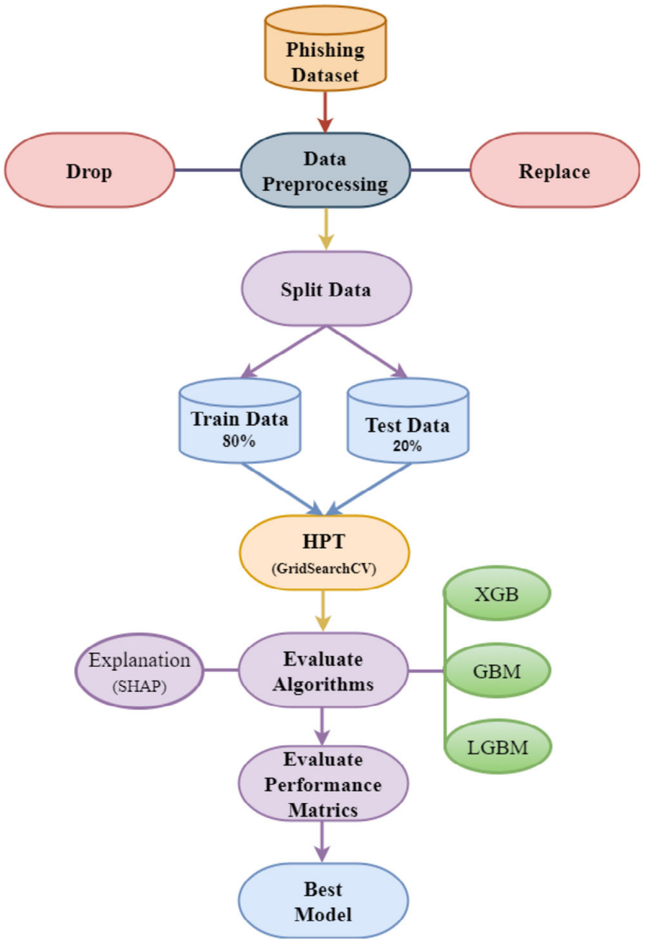

FIGURE 3 | Working procedure of the proposed methodology.

with explainable machine learning techniques like SHAP. Unlike some other datasets, such as those from UCI or Mendeley, this dataset is clean, has no missing values, and is popular in recent phishing detection research. However, it comprises three different types of values, where 1 denotes legitimate, 0 denotes suspicious, and −1 represents a phishing value. There may be an excessive number of missing values and imbalanced data in the dataset, which lowers the accuracy of the results [23]. This data set, however, contained no missing values. The index column has been dropped during data pre-processing because it has no significance for the outcome. Additionally, the dataset output is replaced from −1 to 0, which denotes phishing, because the output of the machine learning classifier can only support 0 or 1. Figure 4 demonstrates the URL dataset's type. There are 6157 real events and 4898 phishing incidents. Since there isn't a huge disparity in the count of the target variable values, this dataset is pretty much balanced.

3.2 | Hyperparameter Tuning

Machine learning uses hyperparameter tuning. The value is set before the model begins the training stage. It enhances the output result with minimal mistakes by limiting a specified loss function [24]. It mostly influences the behaviour of a machine-learning classifier. GridSearch and the cross-validation technique for hyperparameter tuning are implemented in this study to deter-

ALGORITHM 1 | Procedure for predicting XAI phishing URLs

```
Input: Dataset of phishing URLs from Kaggle
Output: Predicted value of XAI phishing URL (Yes or No)
Begin
  1. Load and Preprocess dataset
     • Drop unnecessary columns
     • Replace target values
  2. Split Dataset
     • X = D.drop("Result"), y = D["Result"]
     • Split into (X_train, X_test, y_train, y_test)
  3. Initialize Classifier Models
     • models = [GBM, XGBoost, LightGBM]
  4. For each model in models:
     • Apply GridSearchCV for hyperparameter tuning
     • Train on (X_train, y_train)
     • Predict: y_pred = model.predict(X_test)
     • Evaluate: accuracy, precision, recall, F1-score
  5. SHAP Explainability Analysis
     • explainer = shap.Explainer(model, X_test)
     • shap_values = explainer(X_test)
     • If plots == "bar": shap.plots.bar(shap_values)
     • Else if "beeswarm":
       shap.plots.beeswarm(shap_values)
     • Else: shap.plots.waterfall(shap_values[instance])
  6. Return predictions and SHAP explanations
End
```

mine the best hyperparameters [25]. This experiment uses 10-fold cross-validation. For hyperparameter tuning, we used GridSearch along with 10-fold cross-validation. This approach allowed us to evaluate all possible parameter combinations systematically. We chose this method because it behaves consistently and produces repeatable results. It also fits our goal of explainability, making sure the tuning process stays clear and understandable. The model for GBM in this study was set up with n_estimators = 100 and max_features = 'sqrt', while XGBoost was set with n_estimators = 100 and the default parameters. LightGBM was set up with boosting_type = 'gbdt', n_jobs = 5, silent = True, and random_state = 5. These setups provided a strong baseline and kept things consistent across all models for a fair comparison.

3.3 | Applied Machine Learning Classifiers

This section describes the three gradient-boosting ML classifications in this phishing dataset to evaluate our implementation.

3.3.1 | Gradient Boosting Machine

Gradient boosting machine is a popular progressive learning ensemble technique in learning algorithms that integrates several weak learning approaches to generate an effective anticipatory model in order to increase its accuracy [26]. Gradient boosting

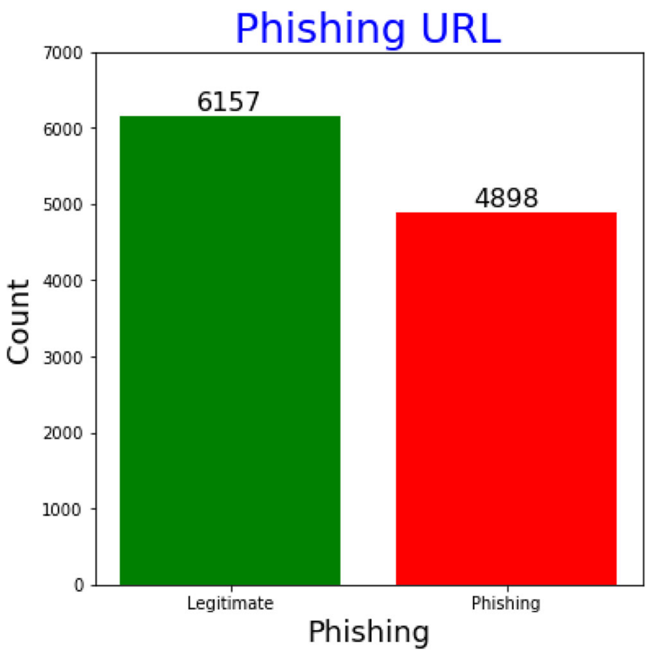

FIGURE 4 | Class distribution of phishing attack detection.

consists of three steps. Among them, the first one is the loss function, which is initially optimized, followed by a prediction from weak learners, and finally, an additive model is introduced in weak learners to lessen the loss function. The loss function of any target column is

$$\mathcal{L}\,(y,\hat{y}) = \sum_{i=1}^{n} \ell\;(y_i,\hat{y}_i) \tag{1}$$

Where y_i is the actual value, $\hat{y}_i$ represents the predicted value and ℓ are the differentiable loss functions

3.3.2 | XGBoost

Extreme gradient boosting, the most recent version of the gradient boosting technique, functions in a manner that is very similar to GBM [27]. It was developed by Chen and Guestrin [28]. This approach systematically incorporates trees that fix the flaws of earlier trees. It functions as a second-order structure of Taylor series augmentation overlying the target function and incorporates further regularization that lessens the model's underfitting or overfitting. In this way, the model's performance is improved. The target function in the context of reduction and regularization is easily stated as given by the XGBoost model's objective function, which is expressed by the following equation [29]

$$\mathcal{L}\,(\phi) = \sum_{i=1}^{n} \ell\,(y_i,\hat{y}_i) + \sum_{k=1}^{K} \Omega\,(f_k) \tag{2}$$

Where ℓ means training loss function, $\Omega(f)$ is the regularization term, and f_k represents individual trees in the ensemble.

3.3.3 | Light Gradient Boosting Machine

LightGBM is actually an upgraded version of the decision tree-based gradient boosting machine and may be used for placement, classification, and a variety of other AI tasks [30, 31]. This classifier employs a histogram-based methodology [32], which splits the ongoing eigenvalues into k time frames and selects divided lines from across the k values. Employing this methodology lowers the cost of storage and computation and improves network connectivity. The expansion of the leaves allows it to process a large amount of data with less inaccuracy. This histogram technique effectively avoids overfitting and has a regularization impact. The simplified formulation for LightGBM method is represented by Equation (3) [33]

$$\mathscr{L}(\mathrm{f}) = \sum_{i=1}^{n} \ell\left(y_i, f(x_i)\right) + \lambda \,||\, f \,||^2 \tag{3}$$

Where ℓ is the loss function, and λ is the regularization coefficient.

3.4 | SHAP (Shapley Additive Explanation)

SHAP is a game-theoretic interpretation approach [31] that may be used to assess the results produced by any ML model. This is model-independent. The computation of feature significance values represents the impact of every attribute on the predictions. SHAP values, which are founded on the Shapley values concept from cooperative game theory, have the ability to exhibit consistency features, local accuracy, and missingness that do not appear conjointly included in different methods [32]. SHAP values can be calculated using the cooperative game theory as follows [31] -

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!\,(|F|-|S|-1)!}{|F|!}\left[f_{S\cup\{i\}}\left(x_{S\cup\{i\}}\right) - f_s\left(x_s\right)\right] \tag{4}$$

where F is the set of all features, S is the subset of features excluding i, and F is the model prediction function.

The additive feature attribution approach is represented by the equation below, where g is specified as a linear function of binary features.

$$g(z') = \emptyset_0 + \sum_{i=1}^{M} \emptyset_i \, z'_i \tag{5}$$

where $z' \in \{0,1\}^M$ works as simplified input features, $\emptyset_i$ is SHAP value for feature i and $\emptyset_0$ is the model's expected output.

SHAP is crucial for boosting algorithms since it provides details about the significance of the characteristics as well as explanations for the predictions provided by these models. Model agnosticism is the primary reason for selecting SHAP for this experiment. These methods do not depend on specific assumptions about the underlying model, which is helpful for model-agnostic analysis. Boosting techniques give importance to features based on how much they help improve the model's accuracy. The relative importance of the features in the boosting method is better understood with the aid of SHAP values. When determining the importance values, SHAP takes into account the complex interactions that boosting algorithms frequently capture in order to increase their predictive performance. It offers a more thorough comprehension of feature importance. Another reason for choosing this approach is that SHAP can offer both local and global explanations for boosting algorithms. To provide clear explanations for boosting algorithms, SHAP values can be visually represented using a variety of plots, including the summary plot, waterfall, and beeswarm. The interpretability of the boosting algorithm is improved by these plots. These are the motivations driving our decision to use SHAP to improve algorithms in this study.

TABLE 1 | Experiment with the proposed system's configuration.

Resource	Details
CPU	Intel Core i3-1005G1 CPU @ 1.20 GHz
RAM	12 GB
GPU	Intel UHD graphics
Software	Anaconda
Language	Python

3.5 | Evaluation Metrics

The confusion matrix was applied to determine the performance metrics. Four metrics, namely true positive (TP), true negative (TN), false negative (FN), and false positive (FP) values, are evaluated to measure each technique's effectiveness. The confusion matrix was calculated using the following equations [34]-

$$\text{Accuracy}\ (\%) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100 \tag{6}$$

$$\text{Precision}\ (\%) = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100 \tag{7}$$

$$\text{Recall}\ (\%) = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100 \tag{8}$$

$$F_{\text{Measure}}\ (\%) = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \times 100 \tag{9}$$

$$\text{Error}\ (\%) = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100 \tag{10}$$

where TP represents the entire number of malicious URLs accurately classified as malware URLs, FN represents the total number of phishing URLs deceitfully defined and detected as legitimate URLs, TN represents the total percentage of valid URLs properly categorized as legitimate URLs, and FP represents the total quantity of correct URLs misclassified as phishing websites.

4 | Results and Discussion

This part outlines the investigations that were carried out and gives the findings in detail. The entire dataset is partitioned into 80:20 segments for both training and testing purposes. The simulated environment was constructed following the specifications listed in Table 1.

TABLE 2 | Evaluation of explainable machine learning method (GBM).

	0	1	Accuracy	Macro avg	Weighted avg
Precision	0.942122	0.942089	0.942108	0.942106	0.942107
Recall	0.954397	0.926755	0.942108	0.940576	0.942108
F1-score	0.948220	0.934359	0.942108	0.941290	0.942057
Support	1228	983	0.942108	2211	2211

TABLE 3 | Evaluation of explainable machine learning method (XGB).

	0	1	Accuracy	Macro avg	Weighted avg
Precision	0.973494	0.983437	0.977838	0.978465	0.977915
Recall	0.986971	0.966429	0.977838	0.976700	0.977838
F1-score	0.980186	0.974859	0.977838	0.977522	0.977818
Support	1228	983	0.977838	2211	2211

TABLE 4 | Evaluation of explainable machine learning method (LGBM).

	0	1	Accuracy	Macro avg	Weighted avg
Precision	0.967949	0.979232	0.972863	0.973590	0.972965
Recall	0.983713	0.959308	0.972863	0.971511	0.972863
F1-score	0.975767	0.969168	0.972863	0.972467	0.972833
Support	1228	983	0.972863	2211	2211

TABLE 5 | Execution time of different ML models.

XML models	Execution time (s)
GBM	0.0061
XGB	0.0085
LGBM	0.0070

The test set was used to acquire the prediction results of the GBM, XGBoost, and LightGBM algorithms. The predictive ability of the model is tested utilizing a range of performance matrices, including F1-score, precision, accuracy, and so on. Classification results of GBM, XGBoost, and LGBM are presented in Tables 2, 3, and 4, respectively. The analysis of the aforementioned tables demonstrates that XGBoost is more accurate than the other classifiers. With an accuracy of 97.28%, LGBM is the second-best classifier in this experiment, while GBM has the lowest performance. The accuracy of this investigation is shown in Figure 5.

XGBoost has greater accuracy than the other classifiers, but its execution time is also higher. Some classifiers may be more accurate because they can identify complex data patterns that require more time to train and make predictions. Additionally, ensemble techniques like GBM, XGBoost, and LightGBM combine many models to increase accuracy, which is another reason for the longer execution time. Table 5 represents the execution time of the applied classifiers of this experiment.

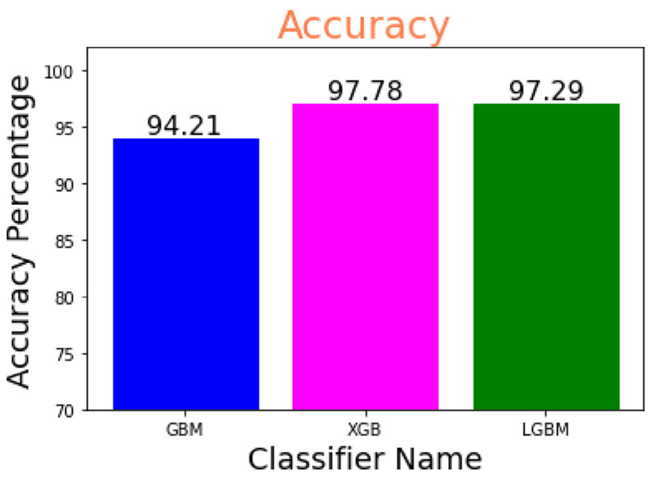

FIGURE 5 | Accuracy results of the GBM, XGB, and LGBM.

Although all three ensemble models, GBM, XGBoost, and LightGBM, share a tree-based boosting structure, they differ in their computational performance both theoretically and in practice. GBM has a time complexity of $O(T.N.F)$, where T is the number of trees, N is the number of data instances, and F is the number of features. XGBoost builds on this with second-order optimization and regularization, resulting in a complexity of about $O(T.D.N.\log N)$, where D is the maximum depth. LightGBM takes scalability even further with histogram-based learning and a leaf-wise growth strategy, achieving a theoretic complexity of

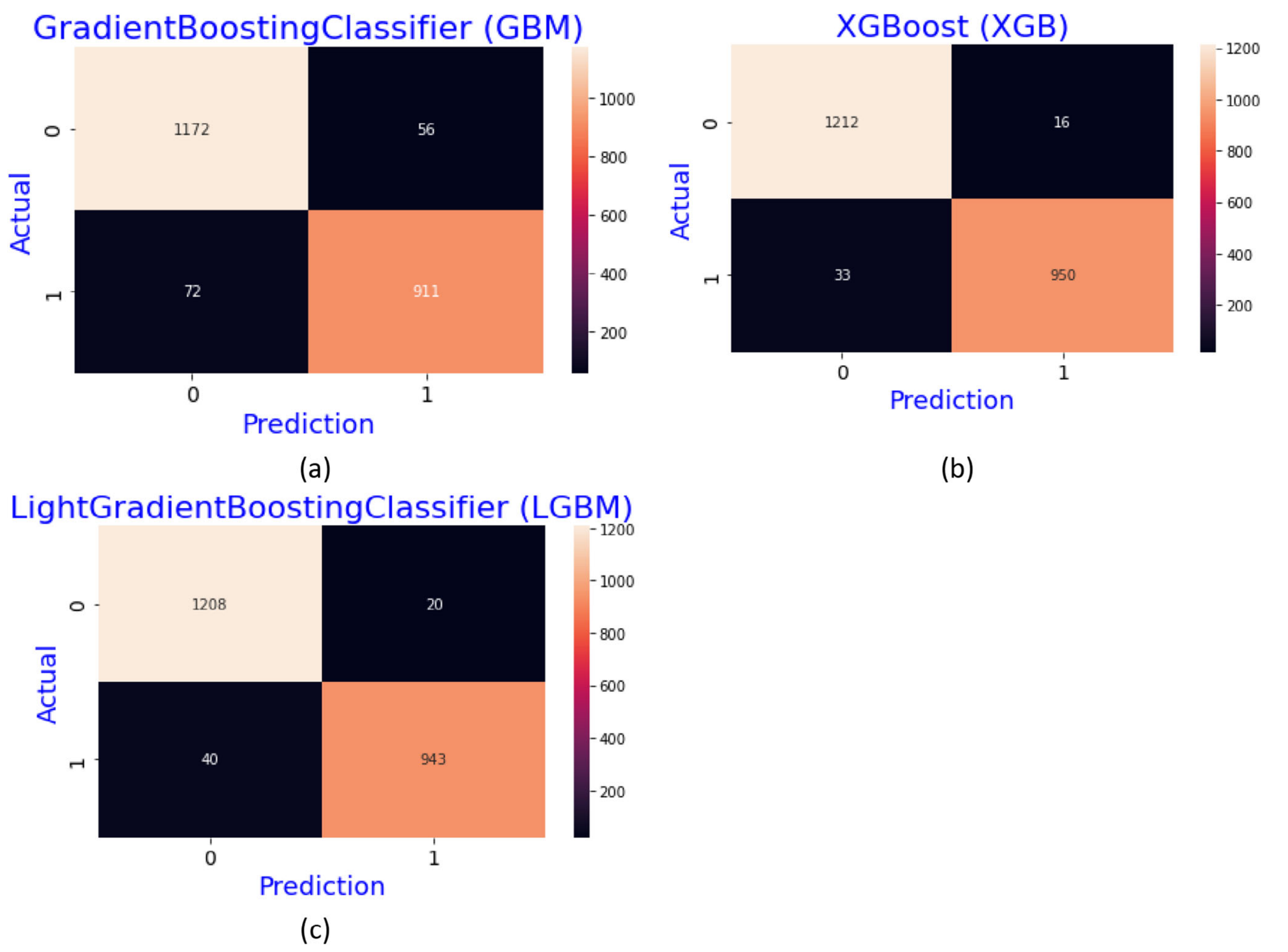

FIGURE 6 | Confusion matrix for (a) GBM (b) XGB, and (c) LGBM.

$O(T.N.\log F)$ However, in our practical evaluation of a relatively small and balanced dataset, GBM recorded the lowest execution time at 0.0061 s. LightGBM followed with 0.0070 s, and XGBoost came in at 0.0085 s. This difference comes from the simplicity and lower overhead of GBM, making it more efficient on smaller datasets, even though it has higher asymptotic complexity. These results suggest that while LightGBM and XGBoost are better suited for larger datasets, GBM may provide faster performance in low-resource or small-scale situations.

To further evaluate the performance of the proposed models, we analysed the confusion matrix for each classifier. The graph of confusion for each classifier is depicted in Figure 6. This gives a detailed breakdown of prediction results in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values are the basis for the evaluation metrics, including accuracy, precision, recall, and F1-score, as defined in Section 3.5. By examining the confusion matrix, we can better understand each model's strengths and weaknesses in classifying phishing and legitimate URLs.

Figure 7 also shows the receiver operating characteristic (ROC) curve, providing a plot of the proportion of true positives versus false positives. It is a graphical representation of a binary classification model's performance in terms of sensitivity and specificity. It is frequently employed to assess and display a classifier's performance.

The ideal classifier point becomes (0,1) if all positive and negative cases have been accurately classified. The vertical axis indicates the detection rate, which is determined by the percentage of phishing attacks discovered. The horizontal axis represents the percentage of false alarms, which is calculated as the proportion of ordinary emails mistakenly identified as phishing attacks. The area under the ROC curve (AUC) is used to quantify accuracy. An area of 0.5 denotes a test that is useless, whereas an area of 1 denotes a superb test. As shown in Figure 7, we obtained an AUC for the training set of 0.988 for GBM, 0.998 for XGB, and 0.873 for LGBM. Additionally, GBM's AUC was 0.986, XGB's was 0.996, and LGBM's was 0.858 for the testing set.

4.1 | SHAP Result Analysis

This part visually represents a few summary plots (Waterfall, Beeswarm) from SHAP to make the machine-learning model more comprehensible and transparent. The impact of every attribute on model forecasts is succinctly summarized here. The representation of the contributions of each feature to the predictions is often shown as a horizontal bar chart. Most

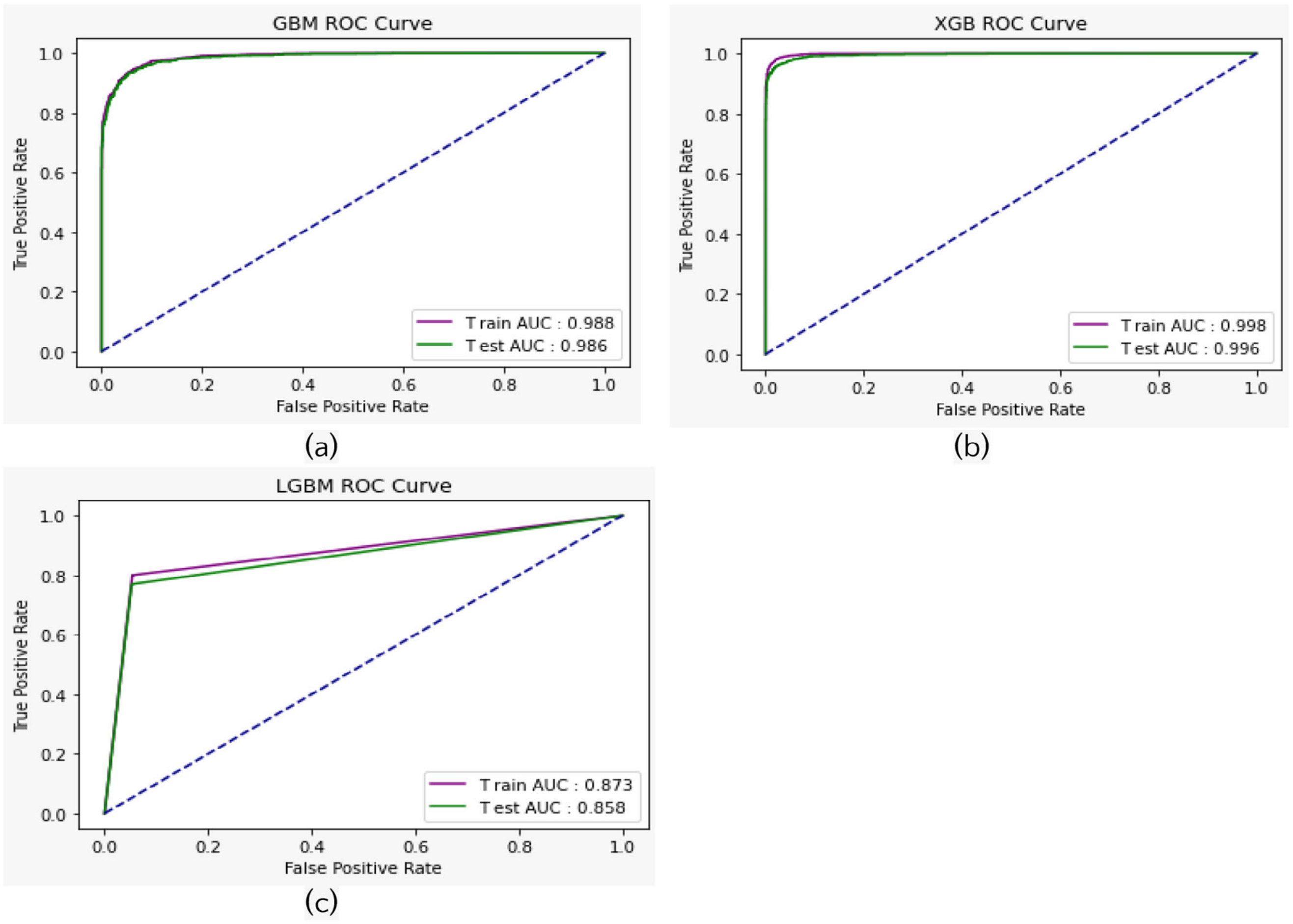

FIGURE 7 | ROC curve (a) GBM, (b) XGB, and (c) LGBM.

often, the features typically appear in a decreasing sequence of magnitude, beginning with the most important. The following Figure 8 represents the importance of the entire dataset for phishing attack detection.

The relevance of every characteristic has been assessed via SHAP values. The SHAP values consider all possible feature combinations to estimate the relative relevance of every attribute in predicting a certain event. The characteristics with higher aggregated actual SHAP values consistently contribute more to the algorithm's estimations. It shows that for predicting phishing attacks, web_traffic contributes the most, at 128%. Links_in_tags contributed 125%, which was the second-highest amount for phishing attack detection. Third-place contributor to this experiment is having_sub_Domain. However, RightClick has the lowest impact on phishing attack detection. The magnitude of each feature's contribution is shown by the blue colour's intensity.

4.1.1 | SHAP Results for GBM

A bar plot, waterfall, and beeswarm for visualizing SHAP results for GBM to clearly show the relative importance of various characteristics in the model's predictions. Figure 9 represents the bar plot for GBM. The features in this figure are presented according to the order of importance to the model's prognosis. The average value on the horizontal axis represents the absolute SHAP measurement for each characteristic.

It is seen that URL_of_Anchor, which contributes an average of +2.17, is the most significant factor. The next most crucial characteristic is SSLfinal_State, which is followed by Prefix_Suffix and web_traffic. By default, it only shows a maximum of ten bars because there is no control over the parameter.

The waterfall plot, which is the most comprehensive display of a single prediction, is the next visualization method. The input variable for the model that "pushes" to forecast a greater phishing attack is represented by the SHAP values in the red bar, and the input variable for the model that "pushes" it to predict a lower phishing attack is represented by the blue bar arrow. Additionally, negative numbers indicate a probability that is less than 0.5. The values of the variables in this specific instance are shown by the grey numbers.

Figure 10 shows that SSLfinal_State's SHAP value is −1.54, followed by web_traffic's SHAP of +1.43, and so on. The total SHAP values in the waterfall plot are equivalent to the variation between the expected number $f(x)$ and the predicted number $E[f(x)]$, which corresponds to the sum of the various SHAP values.

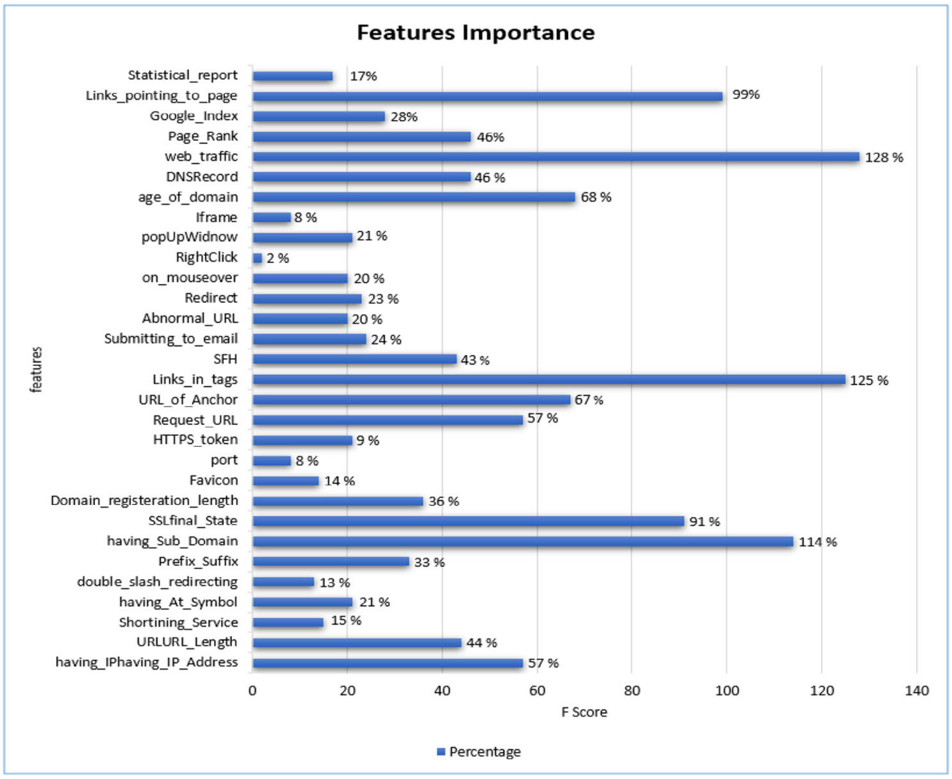

FIGURE 8 | Feature importance determined by SHAP values.

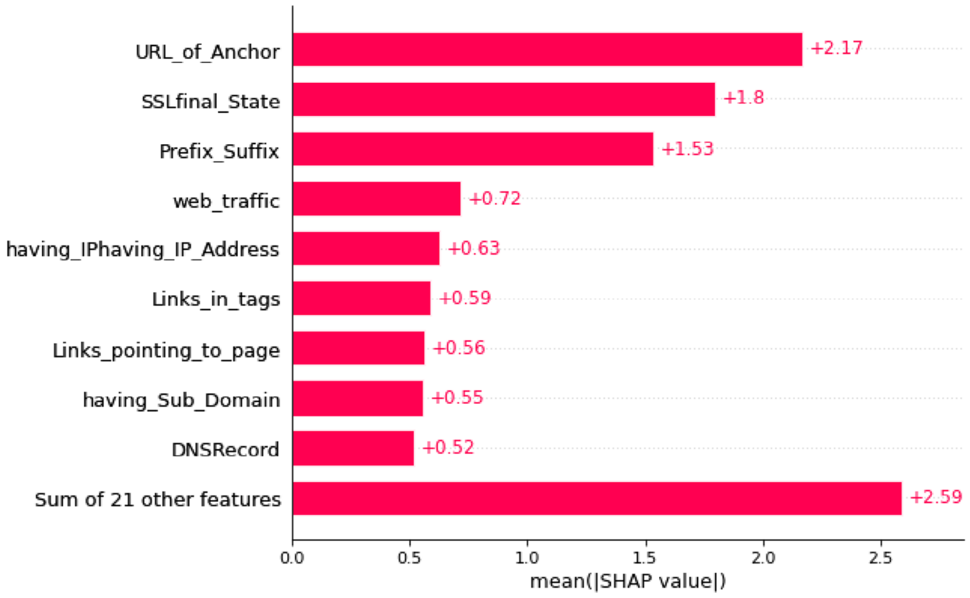

FIGURE 9 | Bar plot for GBM.

Let's verify it-

$$f(x) - E[f(x)] = -1.137 - (-0.86) = -0.277$$

$$\text{Total contribution} = -1.54 + 1.43 + 1.08 - 0.98 - 0.48 - 0.46 + 0.44 - 0.39 + 0.38 + 0.25 = -0.277$$

Another plot, namely the Beeswarm plot, is utilized to better comprehend the significance or contribution of the features to the entire dataset. This graph depicts the link between the underlying values of each characteristic and the model's forecast. The plots of tiny dots each stand for one observation. Figure 11 represents the Beeswarm plot. This graph demonstrates that URL_of_Anchor positively affects phishing attack prediction with a lower value. The lower value of SSLfinal_State leads to a higher chance of phishing attacks. The prediction is largely unaffected by the sum of the remaining 21 features.

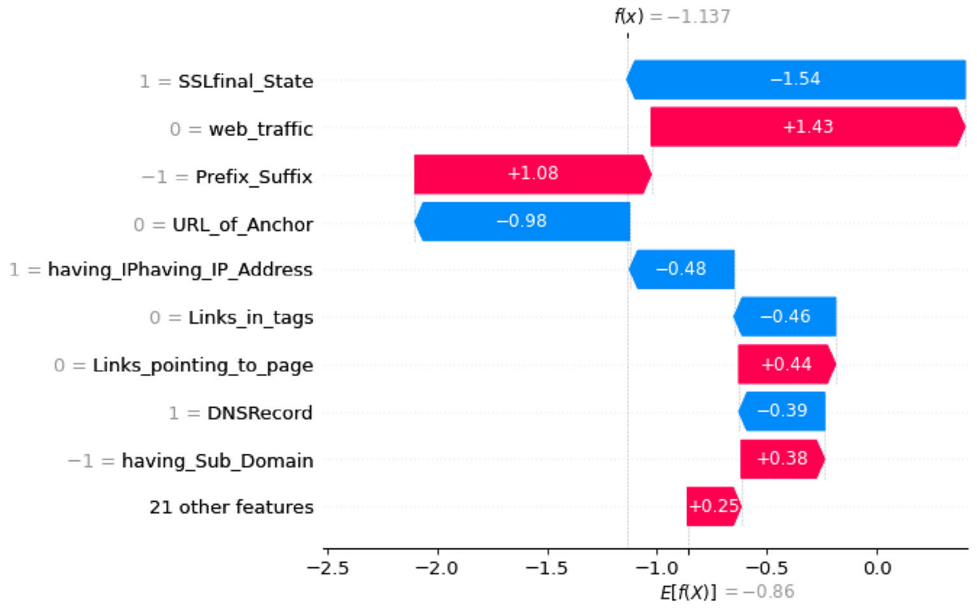

FIGURE 10 | Waterfall plot for GBM.

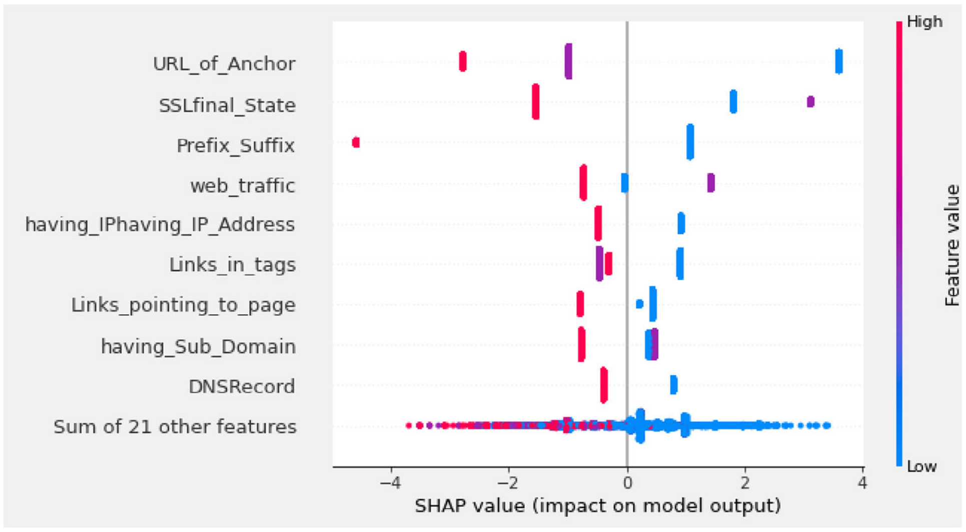

FIGURE 11 | Beeswarm plot for GBM.

4.1.2 | SHAP Results for XGBoost

The value of SHAP is examined in this part to quantify each feature's contribution to the projection produced by the XGBoost approach. The SHAP values produced by XGBoost are unique to each instance and reflect the relevance of the additive feature in that individual instance. A baseline or reference value is utilized to compare every feature's influence on the estimate. The significance of various attributes and their impact on specific predictions can be understood by examining the SHAP values for XGBoost using different plots.

Figure 12 is a bar plot for XGBoost, and it demonstrates that, with a +2.84 value, URL_of_Anchor has the greatest contribution to phishing detection. SSLfinal_State, with a +2.18, is the next significant contributor. Prefix_Suffix is the third most important factor for XGBoost, followed by web_traffic, having_sub_Domain.

Figure 13 shows that web_traffic, with a SHAP value of +1.62, positively affects the prediction. This increases the model's confidence in labelling the URL as phishing. In contrast, SSL_final_State has a SHAP value of −1.56. This indicates a negative effect that lowers the phishing probability. The sum of all SHAP values for this plot is equal to $f(x) - E[f(x)]$.

$$f(x) - E[f(x)] = -1.036 - (-0.542) = -0.494$$

$$\text{Total contribution} = +1.62 - 1.56 - 1.29 + .71 + .47 - 0.39$$
$$- 0.31 + 0.22 - .21 + .25 = -0.49$$

Figure 14 depicts the Beeswarm plot, which demonstrates that URL_of_Anchor has the highest positive impact when its values are low and a substantial negative contribution when its values are high. The likelihood of phishing attacks like GBM increases with a lower value of SSLfinal_State. Lower Prefix_Suffix

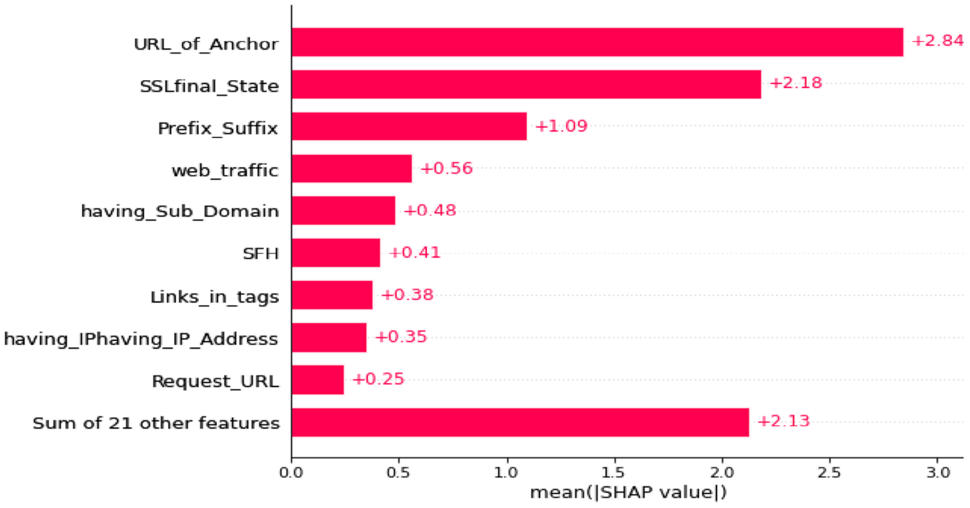

FIGURE 12 | Bar plot for XGBoost.

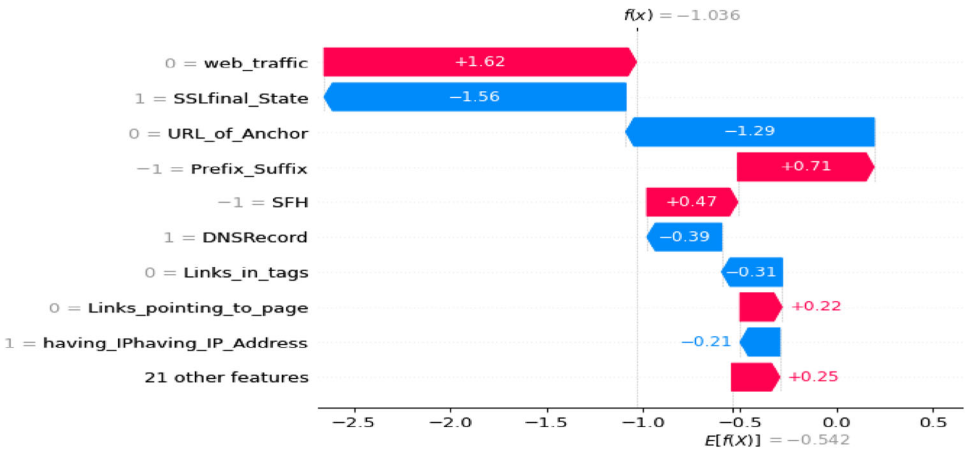

FIGURE 13 | Waterfall plot for XGBoost.

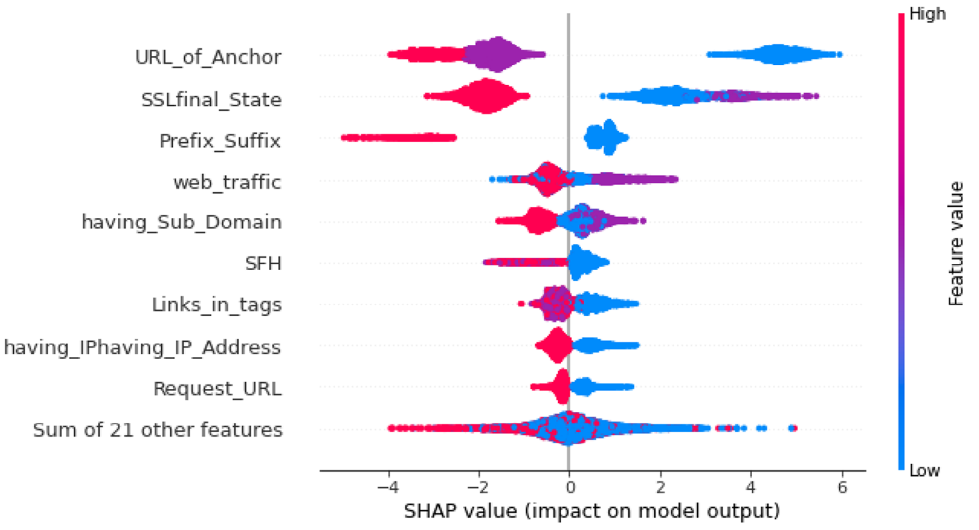

FIGURE 14 | Beeswarm plot for XGBoost.

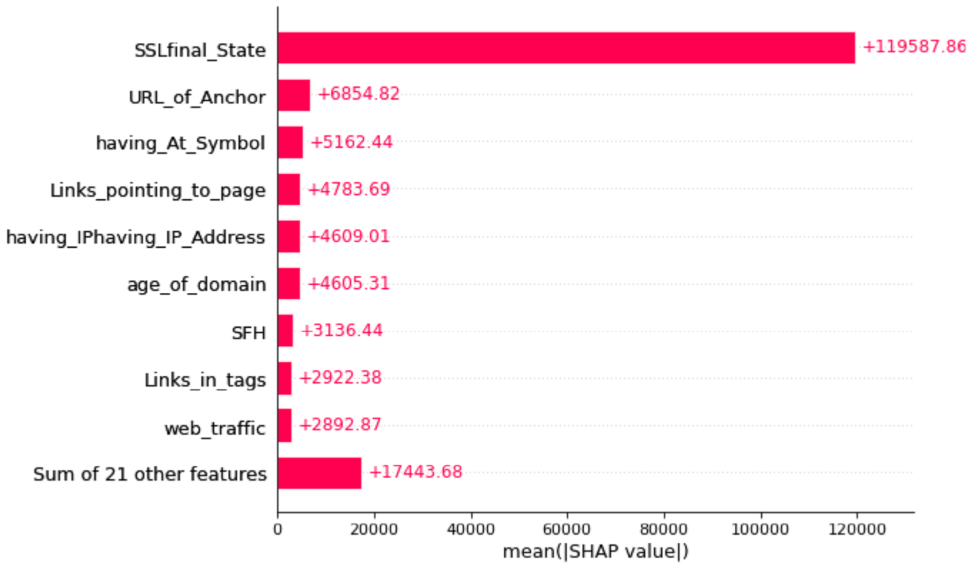

FIGURE 15 | Bar plot for LGBM.

amounts contribute significantly positively to the prediction, whereas high values contribute significantly negatively. Whether their values are high or low, the remaining 21 features barely make any difference to the prediction.

4.1.3 | SHAP Results for LGBM

The SHAP outcomes for LGBM are shown in this section. We utilized the "SHAP" Python module to do SHAP analysis on a LightGBM model since it has functionality for computing SHAP values and analysing the behaviour of the model. When a LightGBM model is used, SHAP values are created to determine the significance of every attribute and its contribution to its forecasts. Figure 15 displays a bar plot that can be used to interpret the SHAP values.

With a value of +119,587.86 compared to the other characteristics, it shows that SSLfinal_State is the more significant feature for LGBM. The second biggest contribution to phishing detection is URL_of_Anchor. The following crucial values are having_at_Symbol, Links_pointing_to_page, and so on.

On the other hand, Figure 16 demonstrates that SSlfinal_State has the most negative influence on prediction, with a SHAP value of −101963.33. After SSlfinal_State, age_of_domain is the most important prediction feature. The $f(x)–E[f(x)]$ for this classifier is: −234414.758 − (−130713.387) = −103,701.371.

$$\begin{aligned}\text{Total contribution} &= -101,963.33 - 4044.84 - 3467.06 - 2895.17\\ &\quad + 2560.54 + 2502.64 + 1764.13 + 1603.59\\ &\quad + 822.56 - 584.43 = -103,701.37\end{aligned}$$

Our experimental results show that it follows the conditions of the waterfall plot.

Figure 17 depicts the Beeswarm plot, which demonstrates that SSLfinal_State has the highest contribution to phishing attack detection. The second most contributing feature is URL_of_Anchor which positively affects phishing attack prediction with a lower value. The lower value of having_At_Symbol leads to a lower chance of phishing attacks.

These SHAP summary plots provide a clear and instructive visual representation of the feature's relevance and contributions. Confidence in the model's intuitiveness and the appropriateness of its conclusions can be established by comparing these insights with domain knowledge.

5 | Discussion

This study demonstrates that by utilizing boosting classifiers, phishing attacks can be predicted well. However, the prediction accuracy was not the same for all the classifiers. The complexity of the method, the number and quality of supported data, and the tuning of the hyperparameters all have a substantial effect on the prediction results of the GBM, XGBoost, and LightGBM algorithms. The results and repercussions of the ML classifiers on the procedure of distinguishing between legal and phishing URLs, as well as SHAP results for each classifier, were shown in the earlier sections. Each classifier's SHAP summary plots demonstrate how a feature influences the prediction. With a positive value, URL_of_Anchor has the most impact on phishing attack detection for GBM and XGBoost, whereas SSLfinal_State has the greatest impact on LGBM. This is evident from the bar plot result. On the other hand, the waterfall figure demonstrates that the GBM, XGBoost, and LGBM are all severely impacted negatively by SSLfinal_State. Additionally, the Beeswarm figure shows how URL_of_Anchor mostly affects phishing attack detection in GBM and XGBoost. To promote transparency, different summary plots are used to display the SHAP value results for this study. XGBoost, on the contrary, performed better than the other employed ML models with accuracy, precision, recall, and F1-score levels of 97.78%, 97.78%, 97.78%, and 97.78%, respectively. This study demonstrates that ensemble tree architecture-based models using Grid search with cross-fold validation achieved a

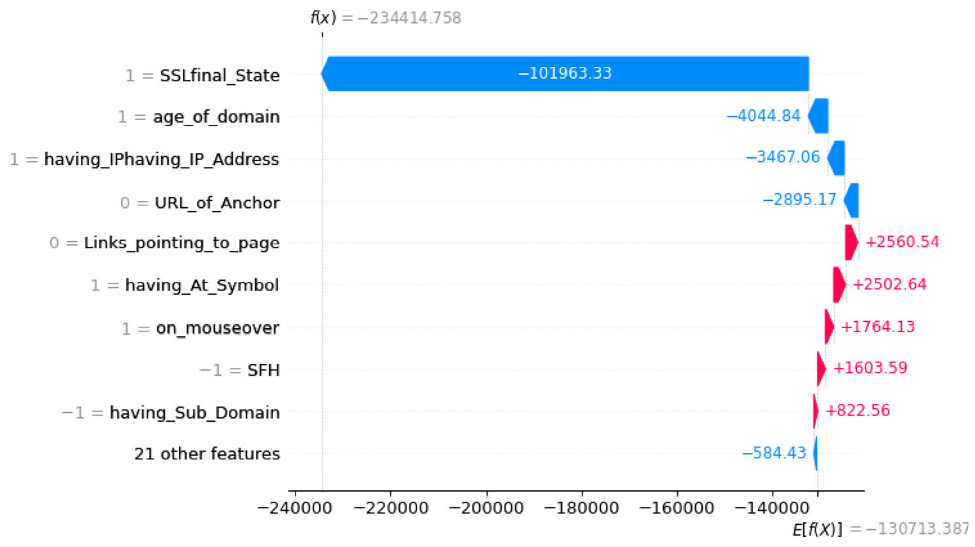

FIGURE 16 | Waterfall plot for LGBM.

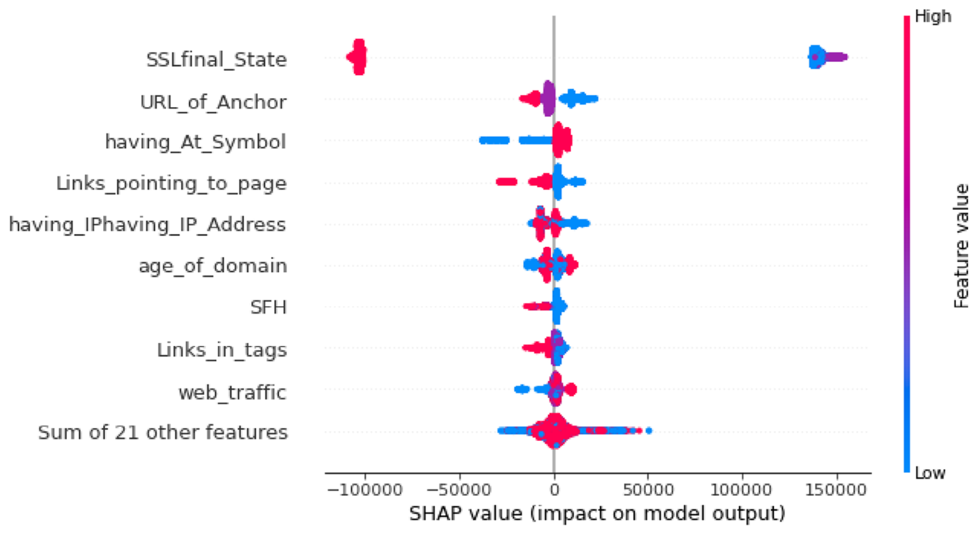

FIGURE 17 | Beeswarm plot for LGBM.

promising outcome on phishing attack detection. Table 6 provides a comparison of the most important recent studies on phishing attack detection. Many existing studies have reached relatively high accuracy using machine learning classifiers, including random forest, K-nearest neighbours, and ensemble models. However, several significant limitations remain. A key problem is the lack of explainability in these models. Earlier research, including that by Vyvaswini et al. [14], Nazmul et al. [15], and Jian et al. [17], mainly aimed for high accuracy but did not include interpretable AI techniques to clarify model decisions. Even in cases where performance was strong, such as an AUC-ROC of 0.96 in Calzarossa et al. [10], the explanation was limited to global feature importance metrics like the Gini index or Lorenz Zonoids. These metrics do not provide instance-level insights, which are essential for real-world implementation and user trust.

Additionally, the problem of generalizability across datasets was also highlighted by certain studies, such as the one by Mia et al. [9], which demonstrated that models trained on one dataset frequently do not perform similarly to another. This demonstrates the shortcomings of feature consistency as well as the significance of robustness testing and dataset diversity. Unbalances between accuracy and recall, or suboptimal recall values, have been found in a number of studies, including those by Nazmul et al. and Karim et al., which indicate that although the models may recognize valid URLs successfully, they may miss numerous phishing URLs, which could be a serious security issue.

Our proposed framework tackles these issues by combining SHAP with high-performance ensemble classifiers like XGBoost, GBM, and LightGBM. This approach not only delivers outstanding predictive performance but also XGBoost+SHAP model achieves 97.78% across all major evaluation metrics. It also offers global and local explanations through SHAP visualizations, including feature importance, beeswarm, and waterfall plots. This two-layer explainability helps stakeholders understand how the model functions overall and why it makes specific predictions.

TABLE 6 | Performance comparison with related studies.

Phishing attack detection			
Author	**Dataset**	**Proposed models**	**Performance**
Shafin [12], 2024	Mendeley	RF	Accuracy: 97.41%
Mia et al. [9], 2024	D1 for Dataset-1 (Vrbančič et al.), D2 for Dataset-2 (Kaggle)	XGBoost for both datasets	Accuracy: Dataset-1: 97.1 %, Dataset-2: 99 %
Babic et al. [13], 2024	Kaggle clickstream dataset	Recurrent Neural Networks	Accuracy:78.2%.
Calzarossa et al. [10], 2023	PhishTank 2 and Tranco	RF	AUC-ROC: 0.96
Jovanovic et al., [11], 2023	Two datasets from Mendeley and one from UCI	XGBoost	Accuracy: Dataset 1: 94.46%, Dataset 2: 95.54%, and Dataset 3: 97.60%.
Vyvaswini et al. [14], 2023	Not mentioned	RF	Accuracy: 95 %
Nazmul et al. [15], 2020	Dataset obtained from Kaggle, consists of 11054 instances and 32 columns.	RF	Accuracy: 97% Precision: 96.89% Recall: 42.16% F-score: 58.74%
Basit et al. [16], 2020	UCI machine learning repository, 11055 instances, and 30 features	K-NN+RFC	Accuracy: 97.33% Precision: 97% Recall: 98.3% F-score: 97.6%
Jian et al. [17], 2019	Phistank dataset, 1 feature, 26580 instance	RF	Accuracy: 97.31% Precision: 93.7% Recall: 92.05% F-score: 92.07%
Sundara et al. [18], 2022	MillerSmiles and Phish Tank	Light GBM	Accuracy: 85.5% Train accuracy: 86.9% Test accuracy: 85.5%
Mahajan and Siddavatam [19], 2018	Phishtank dataset, 36,711 URLs instances, and 16 features	RF	Accuracy: 97.14% False negative rate: 3.14 false positive rate: 2.61
Gaoqing et al. [20], 2020	Phishing dataset and Enron dataset total 1,234,387 instances	RF	Accuracy: 96.72% True positive:1676 true negative:1804
Karim et al. [21], 2016	Kaggle's dataset consisted of 11054 records and 33 attributes	NB	Accuracy: 88.39% precision: 94.92% recall: 83.71% F-score: 88.96%
Proposed	**Kaggle, 11054 instances and 32 columns**	**XGBoost (XML)**	**Accuracy: 97.78% Precision: 97.78% Recall: 97.78% F-score: 97.78%**

This supports clear, fair, and responsible decision-making in cybersecurity.

Moreover, our use of grid search and 10-fold cross-validation improves the generalizability and robustness of the model. By comparing our results with a widely used, balanced dataset from Kaggle with consistent feature engineering, we make sure that our model's performance is not dependent on a particular dataset. This makes our approach both reliable and practical for use in real-world phishing systems. However, our study has some limitations. First, while SHAP provides valuable insights, it can be computationally heavy, especially when used with large-scale models or real-time systems. Second, even though we use a balanced data set to reduce bias, future research could test on imbalanced and real-time streaming data to check flexibility. Lastly, this research mainly focuses on URL-based features. Adding other types of data, like email content or page visual similarity, may further boost detection accuracy and reliability.

6 | Conclusion and Future Work

An XAI-based phishing detection mechanism is proposed to detect phishing attacks. Three classifiers—GBM, XGBoost, and LGBM—are used to implement the established phishing detection technique. Using SHAP, each classifier is described to give details on how each feature influences predictions and how much impact it has. Among the evaluated models, XGBoost showed the best performance, reaching 97.78% for accuracy, precision, recall, and F1-score. The use of SHAP facilitated both global and local interpretability, providing detailed visualizations that clarify how individual features impact model predictions.

By combining powerful predictive models with explainability tools, this approach improves transparency and trust. It supports ethical and responsible AI use in security-sensitive settings. For future work, we plan to broaden the framework by adding a variety of real-world and multilingual phishing datasets to improve generalization and strengthen. We also want to look into deep learning models like convolutional neural networks and transformer architectures to analyse phishing page content and visuals more effectively. In addition, we will integrate several interpretability methods, such as LIME and integrated gradients, to provide better explanations. We will also work on developing lightweight, real-time phishing detection solutions suitable for browser plugins or enterprise email filters. Lastly, we will explore optimization techniques like genetic algorithms and other heuristic methods to enhance hyperparameter tuning and overall model performance.

Author Contributions

Khandaker Mohammad Mohi Uddin: conceptualization, supervision, validation, methodology, writing – original draft. **Nitish Biswas:** resources, methodology, formal analysis, writing – original draft. **Sarreha Tasmin Rikta:** validation, visualization, writing – review & editing. **Md. Nur-A-Alam:** validation, methodology, writing – original draft. **Rafid Mostafiz:** writing – review & editing, resources, conceptualization.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data supporting this study's findings are available from the corresponding author upon reasonable request.

References

1. Q. Cui, G. V. Jourdan, G. V. Bochmann, R. Couturier, and I. V. Onut. "Tracking Phishing Attacks Over Time," in 26th International World Wide Web Conference, WWW 2017 (ACM, 2017), 667–676.

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, https://doi.org/10.1007/s11235-020-00733-2.

3. P. Ghann, E. D. Tetteh, K. Asare Obeng, and M. Elias, "Preserving the Privacy of Sensitive Data Using Bit-Coded-Sensitive Algorithm (BCSA)," *International Journal of Recent Contributions from Engineering, Science & IT (IJES)* 10, no. 04 (2022): 4–16, https://doi.org/10.3991/ijes.v10i04.35023.

4. F. P. E. Putra, U. Ubaidi, A. Zulfikri, G. Arifin, and R. M. Ilhamsyah, "Analysis of Phishing Attack Trends, Impacts and Prevention Methods: Literature Study," *Brilliance: Research of Artificial Intelligence* 4, no. 1 (2024): 413–421.

5. M. Alawida, A. E. Omolara, O. I. Abiodun, and M. Al-Rajab, "A Deeper Look Into Cybersecurity Issues in the Wake of Covid-19: A Survey," *Journal of King Saud University-Computer and Information Sciences* 34, no. 10 (2022): 8176–8206.

6. A. Gramegna and P. Giudici, "SHAP and LIME: An Evaluation of Discriminative Power in Credit Risk," *Frontiers in Artificial Intelligence* 4 (2021): 752558, https://doi.org/10.3389/frai.2021.752558.

7. M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?" Explaining the Predictions of Any Classifier," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM, 2016), 1135–1144.

8. S. M. Lundberg and S.-I. Lee (2017). A Unified Approach to Interpreting Model Predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017) (pp. 4765–4774). arXiv. https://arxiv.org/abs/1705.07874

9. 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 2024 11th International Conference on Networking, Systems and Security, NSysS 2024 (ACM, 2025), 137–145

10. M. C. Calzarossa, P. Giudici, and R. Zieni, "Explainable Machine Learning for Phishing Feature Detection," *Quality and Reliability Engineering International* 40, no. 1 (2024): 362–373, https://doi.org/10.1002/qre.3411.

11. Click Fraud Detection With Recurrent Neural Networks Optimized by Adapted Crayfish Optimization Algorithm. *Journal of Industrial Intelligence* 2(4), 230–239. https://doi.org/10.56578/jii020404

12. 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, https://doi.org/10.1016/j.dsm.2024.08.004.

13. L. Babic, V. Zeljkovic, L. Jovanovic, et al., "Leveraging Metaheuristic Optimized Machine Learning Classifiers to Determine Employee Satisfaction," in International Conference on Multi-Strategy Learning Environment (Springer, 2024), 337–352.

14. T. Vyvaswini, M. r. P. P. N. Rao, B. Kousalya, G. Pallavi, S. Abdullal, and P. Siddartha, "Phishing Website Detection Using Machine Learning," *International Journal of Advanced Research in Science, Communication and Technology* 3, no. 1 (2023): 462–470, https://doi.org/10.48175/ijarsct-8318.

15. M. N. Alam, D. Sarma, F. F. Lima, I. Saha, R. E. Ulfath, and S. Hossain, "Phishing Attacks Detection Using Machine Learning Approach," in Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020 (IEEE, 2020), 1173–1179.

16. A. Basit, M. Zafar, A. R. Javed, and Z. Jalil, "A Novel Ensemble Machine Learning Method to Detect Phishing Attack," in Proceedings - 2020 23rd IEEE International Multi-Topic Conference, INMIC 2020 (IEEE, 2020), 1–5.

17. J. Mao, J. Bian, W. Tian, et al., "Phishing Page Detection via Learning Classifiers From Page Layout Feature," *Eurasip Journal on Wireless Communications and Networking* 2019, no. 1 (2019): 43, https://doi.org/10.1186/s13638-019-1361-0.

18. S. S. Pandiyan, P. Selvaraj, V. K. Burugari, J. Benadit P, and P. Kanmani, "Phishing Attack Detection Using Machine Learning," *Measurement: Sensors* 24 (2022): 100476, https://doi.org/10.1016/j.measen.2022.100476.

19. R. Mahajan and I. Siddavatam, "Phishing Website Detection Using Machine Learning Algorithms," *International Journal of Computer Applications* 181, no. 23 (2018): 45–47, https://doi.org/10.5120/ijca2018918026.

20. G. Yu, W. Fan, W. Huang, and J. An, 2020, June. An Explainable Method of Phishing Emails Generation and Its Application in Machine Learning. In 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (Vol. 1, pp. 1279–1283). IEEE.

21. A. Karim, M. Shahroz, K. Mustofa, S. B. Belhaouari, and S. R. K. Joga, "Phishing Detection System Through Hybrid Machine Learning Based on URL," *IEEE Access* 11 (2023): 36805–36822, https://doi.org/10.1109/ACCESS.2023.3252366.

22. Akashkr, Phishing Attack Prediction Dataset: EDA and Modelling. Kaggle. Available at: https://www.kaggle.com/code/akashkr/phishing-url-eda-and-modelling/input (Accessed: 10 July 2024).

23. S. T. Rikta, K. M. M. Uddin, N. Biswas, R. Mostafiz, F. Sharmin, and S. K. Dey, “XML-GBM Lung: An Explainable Machine Learning-Based Application for the Diagnosis of Lung Cancer,” *Journal of Pathology Informatics* 14 (2023): 100307, https://doi.org/10.1016/j.jpi.2023.100307.

24. N. Biswas, K. M. M. Uddin, S. T. Rikta, and S. K. Dey, “A Comparative Analysis of Machine Learning Classifiers for Stroke Prediction: A Predictive Analytics Approach,” *Healthcare Analytics* 2 (2022): 100116, https://doi.org/10.1016/j.health.2022.100116.

25. D. M. Belete and M. D. Huchaiah, “Grid Search in Hyperparameter Optimization of Machine Learning Models for Prediction of HIV/AIDS Test Results,” *International Journal of Computers and Applications* 44, no. 9 (2022): 875–886.

26. S. Archana and K. Elangovan, “Survey of Classification Techniques in Data Mining,” *International Journal of Computer Science and Mobile Applications* 2 (2014): 65–71, www.ijcsma.com.

27. C. Xu, X. Liu, H. Wang, et al., “A Study of Predicting Irradiation-induced Transition Temperature Shift for RPV Steels With XGBoost Modeling,” *Nuclear Engineering and Technology* 53, no. 8 (2021): 2610–2615, https://doi.org/10.1016/j.net.2021.02.015.

28. T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 2016, 785–794. https://doi.org/10.1145/2939672.2939785

29. I. U. Ekanayake, D. P. P. Meddage, and U. Rathnayake, “A Novel Approach to Explain the Black-box Nature of Machine Learning in Compressive Strength Predictions of Concrete Using Shapley Additive Explanations (SHAP),” *Case Studies in Construction Materials* 16 (2022): e01059, https://doi.org/10.1016/j.cscm.2022.e01059.

30. X. Zheng, IEEE ITSS, and Institute of Electrical and Electronics Engineers, IEEE ISI 2017 : IEEE International Conference on Intelligence and Security Informatics: Security and Big Data : July 22–24, 2017 - Beijing, China, n.d.

31. G. Ke, Q. Meng, T. Finley, et al., “Lightgbm: A Highly Efficient Gradient Boosting Decision Tree,” *Advances in Neural Information Processing Systems* 30 (2017).

32. S. Ranka and V. Singh, 1998, August. CLOUDS: A Decision Tree Classifier for Large Datasets. In Proceedings of the 4th knowledge discovery and data mining conference (Vol. 2, No. 8, pp. 2–8)

33. Y. Liang, J. Wu, W. Wang, et al., “Product Marketing Prediction Based on XGboost and LightGBM Algorithm,” in ACM International Conference Proceeding Series (ACM, 2019), 150–153.

34. I. Markoulidakis and G. Markoulidakis, “Probabilistic Confusion Matrix: A Novel Method for Machine Learning Algorithm Generalized Performance Analysis,” *Technologies* 12, no. 7 (2024): 113.