Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Explainable Soft-Voting Ensemble Framework for Web page Phishing Detection: Integrating ANOVA-Based Feature Selection, Predictive Reliability, and Computational Efficiency

Md. Sharfuddin

sharfuddin.seu@gmail.com

Southeast University

Md. Shakil

Southeast University

Research Article

Keywords: Phishing Attack Detection, Soft-Voting Ensemble, Machine Learning, ANOVA, SMOTE, Real-Time Detection, Explainable Artificial Intelligence, SHAP, LIME

Posted Date: November 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11739983/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Explainable Soft-Voting Ensemble Framework for Web page Phishing Detection: Integrating ANOVA-Based Feature Selection, Predictive Reliability, and Computational Efficiency

Md. Sharfuddin [1]* , Md. Shakil [2]

[1] Department of Computer Science and Engineering, Southeast University, Dhaka, Bangladesh*
sharfuddin.seu@gmail.com

[2] Department of Computer Science and Engineering, Southeast University, Dhaka, Bangladesh
mdshakil3214@gmail.com

Abstract

Cybersecurity threats including phishing attacks that use deceptive websites to steal sensitive information must be fought with accurate and reliable detection mechanisms. Machine learning can be a good solution to detect malicious patterns in the URLs of the websites and differentiate the phishing websites from the legitimate websites. However, existing approaches emphasize prediction performance rather than model robustness, computational efficiency, real-time feasibility, and interpretability. In the present work, we propose an explainable soft voting ensemble framework to overcome these constraints using the Web Page Phishing dataset with 100,077 instances and 19 URL-based features. The methodology includes five-fold cross-validation, selection of top 16 features based on ANOVA F-test, data cleaning, data splitting and smote-based class balancing. The assessment of ten baseline and advanced classifiers led to the selection of CatBoost (a), K-Nearest Neighbors (b) and Random Forest (c) as the top three accuracy-based models for the construction of four soft-voting ensembles. The a+b+c ensemble proposed achieved the highest accuracy of 89.22%, F1-score of 85.55%, AUC of 96.08%, MCC of 0.7715, and the lowest MSE and Brier score of 0.10778 and 0.07498 respectively. Inference latency of 0.386 ms/sample confirmed real-time capability. Training time, inference latency, memory footprint, and serialized model size were measured as operational metrics. In addition, SHAP and LIME enhanced model transparency by offering global and local explanations. Overall, the suggested framework enhances phishing detection in terms of predictability, operational feasibility, explainability and reliability.

Keywords: Phishing Attack Detection, Soft-Voting Ensemble, Machine Learning, ANOVA, SMOTE, Real-Time Detection, Explainable Artificial Intelligence, SHAP, LIME.

1. Introduction

The rapid growth of digital services, online banking and e-commerce has exposed users to the cyber security threats, especially phishing attacks. Phishing is a fraud to obtain sensitive information such as usernames and passwords, financial information or personal data through fake websites that look like real companies [1]. This means that this threat is still present . In the fourth quarter of 2025 , there were 853,244 phishing attacks , according to the Anti-Phishing Working Group (APWG) [2] . With phishing techniques becoming increasingly sophisticated, the need for accurate, reliable and intelligent detection systems is more important than ever.

Traditional phishing detection mechanisms such as blacklisting, whitelisting and heuristic

based mechanisms often fail to detect newly generated phishing URLs and changing attack patterns [3]. Moreover, false negative predictions may cause financial loss and credential theft of users and false positive predictions may disrupt legitimate online activities . These challenges motivate the design of automated detection systems that balance predictive accuracy, computational efficiency, reliability and transparency .

A good solution of ML is to detect discriminative patterns in URL based features and classify the websites as legitimate or phising. Sahingoz et al. [4] showed the efficiency of machine learning algorithms for URL based phishing detection. Traditional classifiers serve as good baselines for comparison, and more sophisticated ensemble models can capture the complex relationship between features and achieve better classification performance. Soft-voting ensembles improve prediction robustness by averaging the predicted probabilities of several classifiers.

But still there are some research gaps. Existing studies tend to focus on traditional classification metrics, with little assessment on probabilistic reliability, computational requirements and practicality for deployment. Recent works have demonstrated the importance of feature optimization, computational efficiency and explainable artificial intelligence (XAI) for the construction of effective phishing detection systems [5], [6]. Therefore, a unified framework integrating strong baseline and advanced models, robust performance and reliability metrics, operational feasibility, and model interpretability remains an important research direction.

To overcome these limitations, this study proposes an explainable soft voting ensemble framework using Web Page Phishing dataset which contains 100,077 instances and 19 URL-based features. The methodology includes data cleaning, dataset splitting, selection of top 16 features using ANOVA F-test, balancing the classes using SMOTE and five fold cross validation. We assess ten baseline and advanced machine learning algorithms. The three best performing individual classifiers are CatBoost (a), K-Nearest Neighbors (b) and Random Forest (c). Then, four soft-voting combinations are built, the a+b+c ensemble being the proposed model.

The proposed ensemble achieves the highest accuracy of 89.22%, F1-score of 85.55%, AUC of 96.08%, MSE and Brier score of 0.10778 and 0.07498 respectively. The inference latency is 0.386 ms/sample, so real-time detection is possible. In addition, SHAP and LIME are used to explain the model globally and locally to improve transparency.

The main contributions of this work can be summarized as follows:

1. Robust Baseline & Advanced Model Evaluation: We will perform an extensive evaluation of ten basic and advanced machine learning algorithms to develop a good benchmark for further experiments and to determine which models are the best in detecting phishing websites.

2. Improved Feature Selection: The best 16 features from the URLs were selected using an ANOVA F-test based technique. This reduces the number of features used and the amount of computation required, yet retains the necessary information needed to accurately detect phishing.

3. Class Imbalance Handling and Robust Validation: In an effort to improve the representation of minority class and to enable effective model performance assessment over multiple data subsets, SMOTE based class balancing and five fold cross validation have been incorporated in the experimental design.

4. Soft-Voting Ensemble Based On Accuracy: Well-performing individual models were selected based on accuracy, and the best performing individual classifiers were CatBoost (a), K-Nearest Neighbors (b), and Random Forest (c). Then four soft-voting ensemble combinations were systematically evaluated and the a+b+c ensemble achieved the highest overall classification accuracy of 89.22%.

5. Comprehensive Performance and Reliability Evaluation: The framework developed will encompass conventional performance indicators (Accuracy, Precision, Recall, and F1 Score) in addition to other performance indicators (AUC, MCC, MSE, Brier Score, and Confusion Matrix Analysis) to evaluate its ability to predict outcomes, misclassify data, and provide accurate probabilities.

6. Feasibility of Practical Implementation & Operational Evaluation: This evaluation investigates the feasibility of the implementation of the ensemble in terms of computational resources (training time, inference latency, RAM size consumed by the model during operation and the file size of the trained model). The proposed ensemble can be used in real-time to detect potential phishing attacks as it takes 0.386 milliseconds per sample for inference.

7. Explainable and Trustworthy Phishing Detection: To improve the transparency, interpretability and trust of the model for cybersecurity applications, both SHAP (global feature importance) and LIME (local prediction explanation) were adopted.

The above contributions collectively build a unified phishing detection system with feature selection/optimization, class balancing, ensemble learning, predictive reliability, operational feasibility and explanatory AI (XAI) to develop viable and reliable cybersecurity systems.

2. Literature Review

2.1. Machine Learning Based Phishing Detection Approach

Machine learning has emerged as an effective approach for detecting phishing websites by identifying malicious patterns in URL structures and website characteristics. Safi and Singh [7] presented a systematic review of 80 research studies which include blacklist-based, heuristic, visual similarity, machine learning and deep learning techniques. Their findings demonstrated an increasing importance of machine learning in the automated detection of phishing. Similarly, Li et al. [8] reviewed emerging phishing detection approaches and identified challenges on feature representation, model generalization, and practical deployment.

Recent studies have explored different feature engineering and classification strategies. Mushtaq et al. [9] presented a feature selection based voting ensemble using two datasets of URLs having 30 and 50 features, respectively. They attained accuracies of 96% and 98%.

Thissuggests the effectiveness of feature optimization and ensemble learning. However, differences in the feature configurations illustrate the significance of a consistent experimental evaluation.

An explainable and lightweight phishing detection framework [10] with PhiUSIIL dataset containing 235,795 URLs integrated Recursive Feature Elimination (RFE), SMOTE, XGBoost and SHAP. The optimized model with four features achieved an accuracy of 85.4% and reduced the computational time by about four times. However, with an F1 score of 0.558, it has limitations regarding the performance of balanced classification. Rashid et al. [11] explored cross-dataset generalization using unsupervised domain adaptation. Their method improves the F1-score by 0.06 on average across all evaluated scenarios, suggesting that traditional models might degrade in performance when evaluated on independently collected datasets.

Ghalechyan et al. [12] studied deterministic and probabilistic neural networks on open source and production-oriented phishing URL datasets. The study they conducted demonstrated the significance of detection and probability estimation based on neural networks. The integration of statistical feature selection, ensemble optimization and global-local explanations, however, is an important research opportunity.

2.2. Feature Selection and Ensemble Learning

Feature selection reduces dimensions, eliminates redundant information, and improves computational efficiency. Jain et al. [13] combined PhishNet 1.0 with Boruta feature selection, Optuna hyperparameter optimization, and stacking ensemble learning. Their optimized stacking model reached an accuracy of 96.15%, precision of 96.45%, recall of 96.68%, and F1-score of 96.56%. The key contribution was stacking optimization, not systematic evaluation of multiple soft-voting configurations, although the framework showed very good predictive performance.

In [14] Khan et al. presented a Bayesian-optimized explainable stacking framework on a dataset of 247,950 URLs and 41 structural and lexical features. Their approach included SHAP-guided feature selection, 12 classifiers, Optuna optimization, and five-fold cross-validation. The optimized ensemble produced 96.75% accuracy, 99.47% AUC and Brier score of 0.0246. However, it uses a stacking architecture, which is different from the heterogeneous soft-voting strategy studied in this paper.
These studies demonstrate the effectiveness of feature selection and ensemble learning. But also the ensemble performance depends on the diversity of its base classifiers and on the quality of the predicted probabilities. It is therefore still meaningful to test different soft-voting combinations with complementary classifiers.

2.3. Operational Efficiency and Explainable AI

However, operational efficiency is crucial to the deployment of phishing detection systems in the real-world cybersecurity environment. Dandotiya et al. [15] proposed a browser extension based on machine learning for real-time phishing detection, showing a practical implementation of automated detection. Their work highlights the need for browser security applications to incorporate classification models. However, predictive reliability, memory requirements and full explainability remain to be explored.

Explainable Artificial Intelligence (XAI) has emerged as an important aspect to understand

complex classification decisions. In [16] Panthi et al. compared SHAP and LIME in explaining boosting-based phishing URL detection models with 11,054 instances and 30 features. Their LightGBM model achieved 97.60% accuracy and SHAP and LIME explained it globally and locally in a complementary way. The study emphasizes the importance of model transparency . However , the main focus was on interpretation of boosting models , rather than an entire ensemble and operational evaluation .

Prasad and Chandra [17] proposed PhiUSIIL, a large scale framework for phishing URL detection using similarity indices and incremental learning. Their research looked into different URL characteristics and adaptive detection. Aljofey et al. [18] studied the combination of URL and HTML features to enhance the detection of phishing websites. The approaches show the significance of feature diversity for detecting sophisticated phishing attacks.

Earlier research also set important foundations. Hannousse and Yahiouche [19] studied benchmark datasets and experimental practices used in machine learning based phishing detection. Sahingoz et al. [20] used lexical and natural language processing features in the evaluation of seven machine learning algorithms, with Random Forest obtaining an accuracy of 97.98%. Xiang et al. [21] proposed a feature-rich machine learning framework named CANTINA+. Marchal et al. [22] developed a streaming analytics based phishing detection named PhishStorm. Further, Awasthi and Goel [23] studied base and ensemble classifiers by cross validation. In general, these studies demonstrate the progression of phishing detection from traditional feature engineering, through adaptive and ensemble based approaches, to explainable approaches.

2.4. Research Gap and Proposed Study

The literature review reveals remarkable progress in machine learning for phishing detection, feature optimization, ensemble learning, cross-dataset evaluation, and explainable AI. But there are several methodological options. First, a systematic comparison of strong baseline and advanced classifiers using multiple soft-voting combinations needs to be further investigated. Second, class balancing combined with statistical feature selection can effectively improve feature relevance and minority class representation. Third, accuracy alone cannot sufficiently describe the reliability of prediction; hence, MCC, MSE and Brier score offer complementary perspectives for evaluation. Fourth, we assess computational feasibility based on training time, inference latency, memory footprint, and serialized model size. Finally, the combination of SHAP and LIME improves global and local interpretability of the model.

The present study proposes an explainable soft-voting ensemble framework to address these opportunities using Web Page Phishing dataset with 100,077 instances and 19 URL-based features. The framework includes data cleaning, splitting of dataset, selection of top 16 features based on ANOVA F-test, class balancing based on SMOTE, and five-fold cross-validation. Ten baseline and advanced machine learning classifiers are tested, and then four soft-voting combinations. The best performing combination is CatBoost, K-Nearest Neighbors and Random Forest with 89.22% accuracy, 85.55% F1-score, 96.08% AUC and 0.7715 MCC. Furthermore, we use MSE and Brier score to measure the prediction reliability and training time, inference latency, memory footprint and serialized model size to measure the computational feasibility. SHAP and LIME are used for global and local explanations respectively. Therefore, the present study aims at developing a unified framework that can integrate the predictive effectiveness, feature optimization, probabilistic reliability, operational feasibility and explainability.

3. Methodology

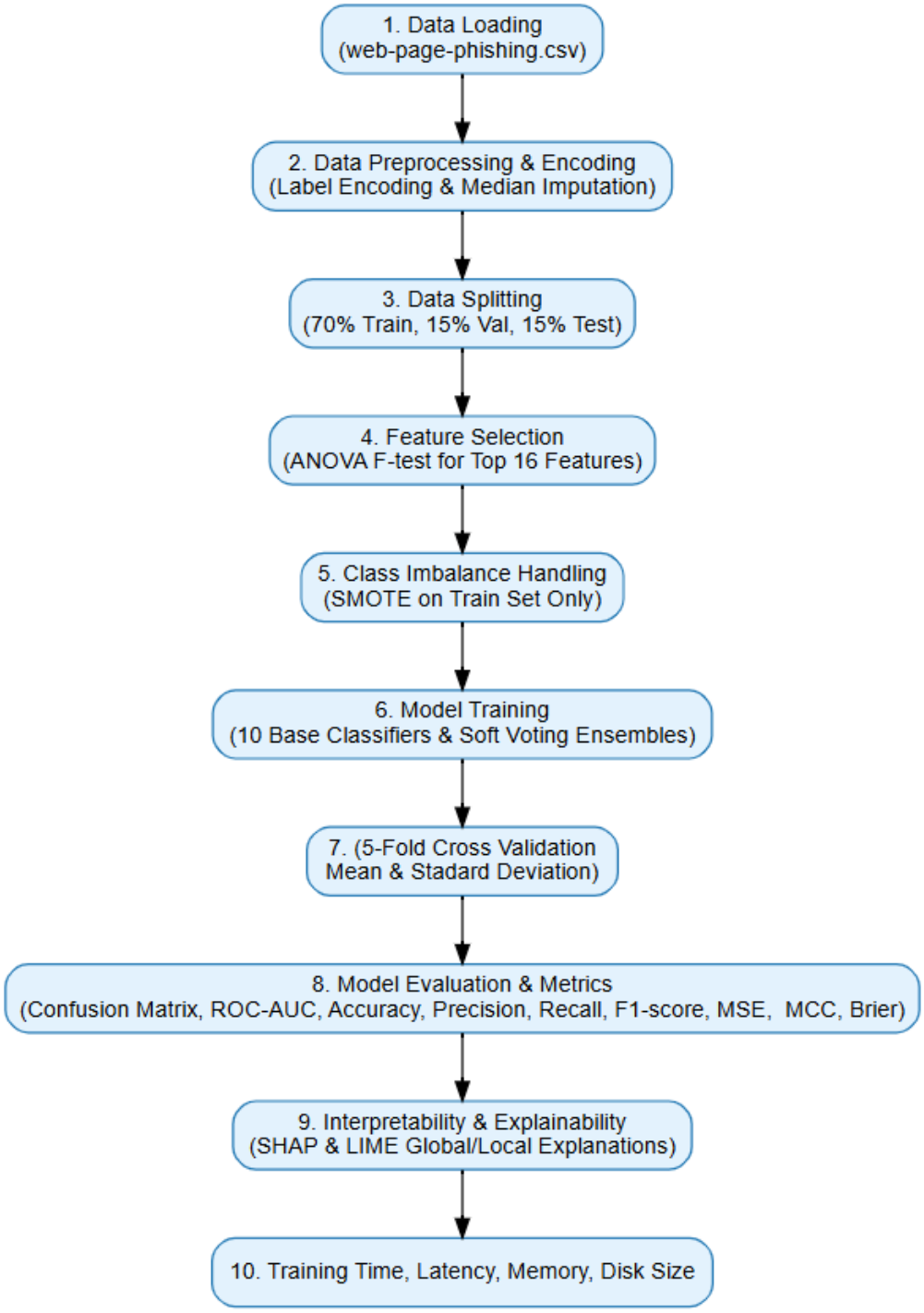

Figure 1. Proposed Machine Learning and Explainable AI (XAI) Framework for Web Page Phishing Detection

3.1 Dataset

This dataset is already available in Kaggle for research purpose only. This dataset consists of clinical and laboratory information of patients with liver cirrhosis. It comprises of demographic, biochemical and clinical characteristics that help in determining the stage of liver cirrhosis.

Link: The dataset were Collected From Kaggle. The data contains 100077 observations with 20 attributes [25].

Class: The dataset has a phishing attribute which is the target variable that denotes the phishing attack .

Features: The dataset contains several clinical attributes: url_length, n_dots, n_hypens, n_underline, n_slash, n_questionmark, n_equal, n_at, n_and, n_exclamation, n_space, n_tilde, n_comma, n_plus, n_asterisk, n_hastag, n_dollar, n_percent, n_redirection, phishing

3.2 Data Processing

3.2.1 Data Encoding

As machine learning algorithms work on numbers, categorical are changed into numbers by using suitable encoding. The simplest approach to handle categorical features is converting them to numbers and one-hot encoding. This method allows the models easy manipulation of relational information and improves prediction accuracy as well.

3.2.2 Feature Selection

Then, relevant features that are associated with the progression of liver cirrhosis were selected to develop a novel model. The machine learning algorithms were trained using the selected independent variables and the Stage attribute was treated as target variable. 16 feature were selected out of 20 feature.

3.3 Data Splitting & Balancing

Models were evaluated with minimal bias from the data by splitting the dataset into three subsets;
Train Set: It is 70% data used to train the model.
Validation Set : 15% of the data used for model selection and hyperparameter tuning.
Test Set: The remaining 15% is set aside for estimating the performance of the model.

Training shape before SMOTE: (70053,16)
Training shape after SMOTE: (89200, 16)
Validation shape: (15012, 16)
Testing shape: (15012, 16)

3.5 Machine Learning Models

3.5.1 Base Models

Material and Methods In the present study, nine machine learning algorithms were used to classify
the stages of liver cirrhosis:

Naïve Bayes (NB)
K-Nearest Neighbors (KNN)
Logistic Regression (LR)
Decision Tree (DT)
Random Forest (RF)
Adaboost
Support Vector Machine (SVM)
XGBoost
LightGBM
CatBoost

The models were assessed using the below metrics:
Accuracy
Precision
Recall
F1-Score
Mean Squared Error (MSE)
AUC
MCC
Brier Score

We compared the performance of all models to identify the best classifier.

3.5.2 Ensemble Models

The three best performing models with machine learning were used to build ensemble models. To enhance the predictive performance, different combinations were generated:

Ensemble Model (a+b)
Ensemble Model (b+c)
Ensemble Model (a+c)
Ensemble Model (a+b+c)
where:

a = Best-performing model
b = Second-best-performing model
c = Third-best-performing model

In this study, similar performance metrics were utilized in developing and analysing the performance of this ensemble approach against single classifiers to find out which prediction framework will provide optimal outcome as a classifier.

3.6 Model Description

Naïve Bayes (NB)
Naive Bayes is a supervised learning method which works on the probabilistic concept of Bayes theorem. It assumes feature independence and can work efficiently using it for classification in more medical prediction tasks.

K-Nearest Neighbors (KNN)
K-Nearest Neighbors (KNN) — is an instance-based learning algorithm which we use to classify any given sample based on the class with highest majority in it nearest neighbors. It uses similarity in patient records to predict stages of cirrhosis.

Logistic Regression (LR)
Logistic Regression is the supervised classification algorithm that estimates a probability of being a class. In this paper, it predicts the stages of liver cirrhosis from clinical and laboratory findings.

Decision Tree (DT)
Decision Tree is a type of rule-based model that splits the data on branches based on unit tests for one feature or another. The way it is structured allows predictions to be understandable and interpretable.

Random Forest (RF)
Random Forest>> It is trained on more than one decision trees and takes an aggregate of the outputs in order to make the classification as a whole.Edge oriented: Random forests are simply bagging and averaging with/without replacement. It is highly used for applications likes medical diagnosis.

AdaBoost
Boosting algorithms, such as AdaBoost, simplify this process by allowing the user to fix a learning method and only have it concentrate on samples that are misclassified.

Support Vector Machine (SVM)
SVM builds the optimal hyperplane which maximized class separation. Well-suited for complex decision boundaries and can handle high-dimensional medical datasets.

XGBoost
XGBoost is an improved version of the Gradient Boosting algorithm, with higher processing power and a top-notch predictive capability. This models the simple dependencies among clinical variables well.

LightGBM
A gradient boosting framework that uses tree-based learning algorithms for training. It works on massive datasets and makes accurate predictions.

CatBoost
A high-performance gradient boosting algorithm that natively handles categorical features and uses ordered boosting to prevent overfitting.

3.7 Evaluation Metrics

To evaluate the classification performance of the models, confusion matrix components and standard evaluation metrics were evaluated.

True Positive (TP)
True Negative (TN)
False Positive (FP)
False Negative (FN)

We used these indicators to calculate Accuracy, Precision, Recall, F1-Score and ROC-AUC for a better evaluation of the model.

Accuracy:

Accuracy How many of the affirmative predictions (phishing) were actually correct .

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

Precision:

Of all the predictions, how many did the model get right.

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

Recall:

Recall: The percentage of actual positive cases (phishing) that were correctly identified by the model.

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

F1-Score:

F1-score: A way to combine precision and recall into one number by taking the harmonic mean.

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

MSE (Mean Squared Error)

Average of the squared differences between the predicted values and the true values

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

TPR / Sensitivity:

The True Positive Rate (TPR) is the fraction of actual positives that are correctly identified as positives by the model . This test demonstrates how good the model is at finding all of the real threats.

$$\text{TPR} = \frac{TP}{TP + FN}$$

False Positive Rate (FPR):

FPR stands for False Positive Rate. This is the number of cases where the model predicts a positive case when it is actually a negative case. It counts how many safe things are misidentified as threats.The True Positive Rate (TPR) is the ratio of correctly predicted positive observations to all actual positives. This test demonstrates how good the model is at finding all of the real threats.

$$\text{FPR} = \frac{FP}{FP + TN}$$

Specificity:

Where specificity is the ability of a model to correctly identify legit websites and not phishing attacks. High specificity increases the practical reliability of the model as misclassification of a legitimate safe website as a phishing threat (false positive) interferes with the user's access. In contrast, the correct clearance of safe sites helps in building user trust in the detection system.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

AUC, or area under the ROC curve:

The area under the ROC curve, usually abbreviated as AUC gives a full evaluation of the model's performance in discriminating among phishing classes. AUC values approaching 1 suggest good discrimination ability and strong class prediction ability, whereas lower values indicate poor predictive performance.

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) \cdot d(\text{FPR})$$

Practical Implementation of AUC:

We have a ROC curve because multiple thresholds can be used to classify binary predictions into one class or another, and the AUC gives an idea of how well our model performs in general across many different threshold values. It then gives a common goal measure to compare different models over the data set for detecting Phisihing.

$$\text{AUC} = \frac{1}{2}\sum_{i=1}^{m-1}(\text{FPR}_{i+1} - \text{FPR}_i) \times (\text{TPR}_{i+1} + \text{TPR}_i)$$

MCC (Matthews Correlation Coefficient):

MCC (Matthews Correlation Coefficient) is a balanced measure which can be used to evaluate the quality of the binary classification on a scale from -1 to +1. A perfect prediction is represented by +1, a random guess is represented by 0 and a total disagreement is represented by -1.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Brier Score

Brier Score is a measure of the accuracy of probabilistic predictions , and is defined as the mean squared difference between predicted probabilities and the actual outcomes . It ranges from 0 to 1 , where 0 is a perfect prediction .

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$$

4. Results

4.1 Experimental Setup

All the code for preprocessing data, model training and evaluation were written in Python 3 on the Google Colaboratory (Colab) cloud environment, on standard cloud-hosted virtual CPU/GPU compute instances and system RAM. The software stack was built from commonly used open-source machine learning and data science libraries:

Preprocessing of Data and Statistical Analysis: The data was preprocessed and the ANOVA F-test statistical analysis was carried out using Pandas, NumPy and SciPy.

Ensemble Frameworks and Machine Learning: The base classifiers and soft-voting ensembles were trained using Scikit-learn, CatBoost, XGBoost and LightGBM.

Handling Unbalance: We used the Synthetic Minority Over-sampling Technique (SMOTE) with imblearn (which stands for "imbalanced learn").

Explainability and Visualization: SHAP (SHapley Additive Explanations), LIME (Local Interpretable Model-agnostic Explanations), Matplotlib and Seaborn were used for the explainability of the models and visualization of the results.

4.2. Data Pre-processing and Feature Selection

The proposed framework for phishing detection is evaluated against the Web Page Phishing dataset which has 100,077 instances and 19 URL based features. Data were cleaned and preprocessed to ensure consistency and suitability for subsequent machine learning experiments. For model development and independent evaluation of the performance, the dataset was divided into training, validation and testing subsets.

The F-test of Analysis of Variance (ANOVA) was used for dimensionality reduction and selection of the most informative attributes. The original feature set was reduced to the top 16 features based on their statistical relationship with the target variable. The rationale behind this reduced representation was to eliminate less informative attributes, but retain relevant characteristics that can be used to differentiate phishing websites from legitimate ones. Also, the Synthetic Minority Oversampling Technique (SMOTE) was applied to deal with the class imbalance issue in the training data. SMOTE generates synthetic instances of minority class based on the characteristics of the neighbouring observations, so that classifiers can learn the minority class patterns better . In this preprocessing strategy, ten baseline and advanced machine learning classifiers and four soft-voting ensemble combinations were evaluated.

4.3. Confusion Matrix Analysis

Confusion matrices were also produced for more in-depth analysis of the classification behaviour of both individual and ensemble models. In each matrix there are four outcomes: True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP). In this experiment, the negative class consists of legitimate websites and the positive class consists of phishing websites.

- **True Negative (TN):** Legitimate websites are identified as legitimate correctly.
- **False Positive (FP):** Real websites classified as phishing.
- **False Negative (FN):** Phishing sites are incorrectly classified as legitimate.
- **True Positive (TP):** Websites correctly identified as phishing.

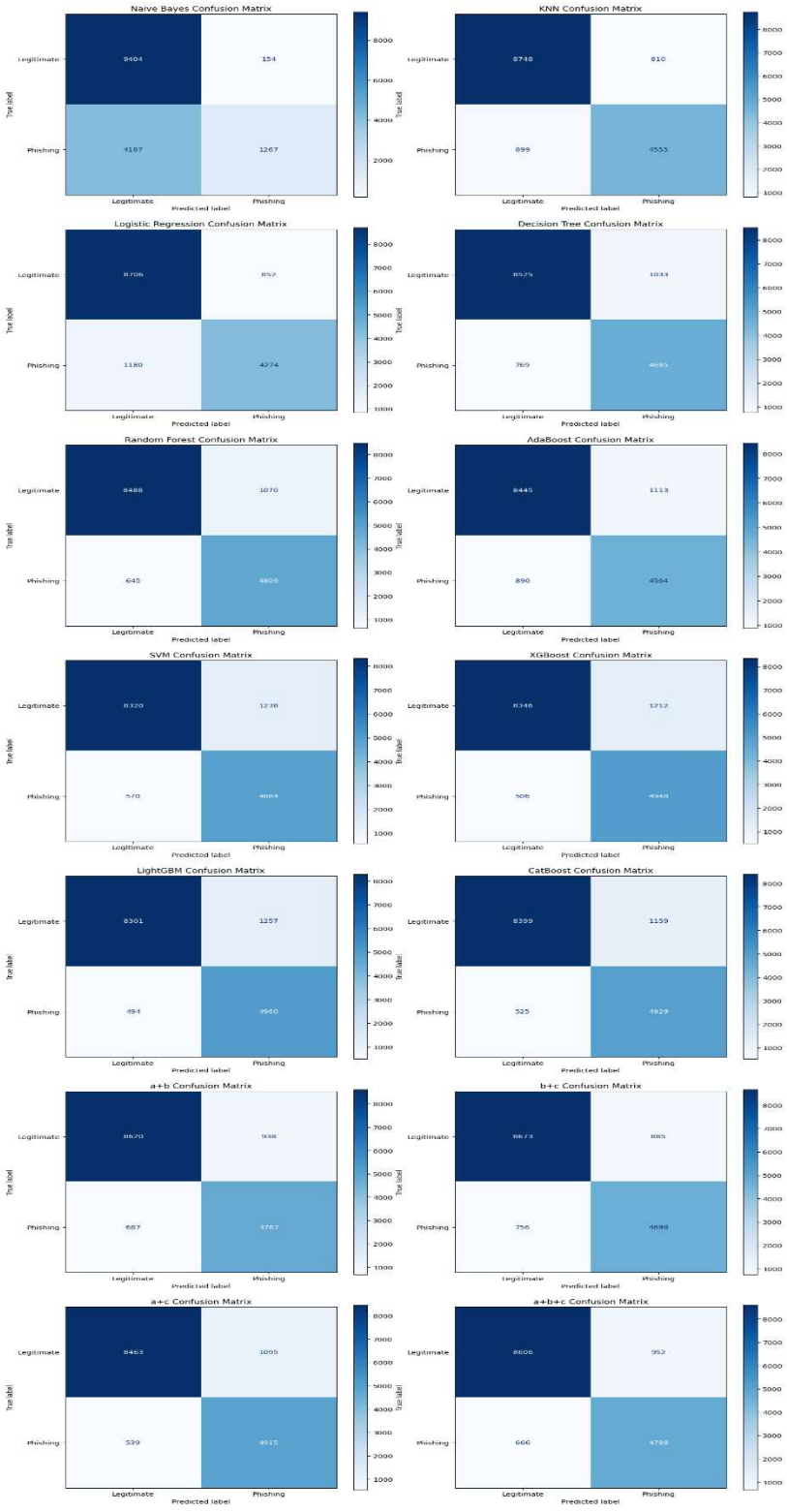

Figure 2. Confusion Matrix of all Models (Base & Ensemble)

Figure 2. contains the confusion matrices for an in-depth evaluation of the classification accuracy of ten individual machine learning algorithms and four voting ensemble combinations

to differentiate legitimate from phishing websites. Regarding the individual classifiers, Naive Bayes classified correctly 9,404 legitimate and 1,267 phishing websites but wrongly classified 154 legitimate websites as phishing and 4,187 phishing websites as legitimate. KNN correctly classified 8,748 legitimate websites and 4,555 phishing websites. It produced 810 false positives and 899 false negatives. Logistic Regression correctly classified 8,706 legitimate websites and 4,274 phishing websites. It produced 852 false positives and 1,180 false negatives. Decision Tree was able to get 8,525 true negatives and 4,685 true positives. It had 1,033 false positives and 769 false negatives. The Random Forest correctly classified 8,488 legitimate websites and 4,809 phishing websites, with 1,070 false positives and 645 false negatives. Similarly, AdaBoost had 8,445 true negatives and 4,564 true positives, with 1,113 false positives and 890 false negatives. The SVM correctly classified 8,320 legitimate and 4,884 phishing sites with 1,238 false positives and 570 false negatives. Among the boosting-based classifiers, XGBoost correctly classified 8,346 legitimate and 4,948 phishing websites, but observed 1,212 false positives and 506 false negatives. LightGBM correctly classified 8,301 legitimate and 4,960 phishing websites, and generated 1,257 false positives and 494 false negatives, while CatBoost correctly classified 8,399 legitimate and 4,929 phishing websites, and generated 1,159 false positives and 525 false negatives. The voting ensemble models, a+b combination, correctly classified 8620 legitimate and 4767 phishing websites. There were 938 false positives and 687 false negatives. The combination of b+c produced 8,673 true negatives, 4,698 true positives, 885 false positives and 756 false negatives. The a+c ensemble successfully identified 8463 legitimate and 4915 phishing websites, with 1095 false positives and 539 false negatives. The result of the a+b+c combination classified 8,606 legitimate and 4,788 phishing websites correctly with 952 false positives and 666 false negatives. To summarise, the confusion matrix results indicate that XGBoost, LightGBM and CatBoost demonstrated good phishing detection performance individually, and the a+c ensemble obtained the highest total correct predictions among the models evaluated. The results suggest that the models examined offer different trade-offs between correctly identifying phishing websites and minimising the classification of legitimate websites as phishing.

The proposed three-model ensemble produced the following confusion matrix:

Table 1. Confusion Matrix of the Proposed a+b+c Ensemble

Actual Class	Predicted Legitimate	Predicted Phishing
Legitimate	8,606	952
Phishing	666	4,788

Table 1. shows, The ensemble proposed correctly identified 4,788 phishing websites and 8,606 legitimate websites. Meanwhile, 952 genuine cases are wrongly classified as phishing and 666 phishing cases are wrongly classified as genuine. The accuracy rate was about 89.22%, with 13,394 out of 15,012 testing instances being correctly classified.

The false negatives are of particular importance from the cybersecurity point of view as they represent the phishing websites that were not detected. The proposed ensemble had 666 false-negative cases which means some phishing cases are still hard to identify. This combination has the highest accuracy among the evaluated combinations. However, the number of false negatives is higher than the a+c ensemble, which correctly identified more phishing instances.

Hence, the analysis of the confusion matrix shows the importance of assessing the phishing detection systems not only by their accuracy, but also by their recall and false-negative rates.

4.4. ROC Curve Analysis of Individual Classifiers

1. Individual Machine Learning Models: ROC-AUC Performance

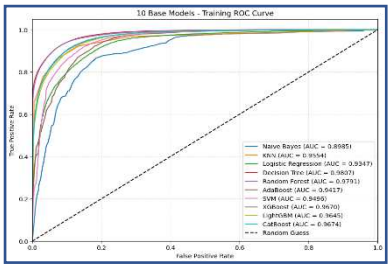

Fig 3. Training ROC for 10 Base Model

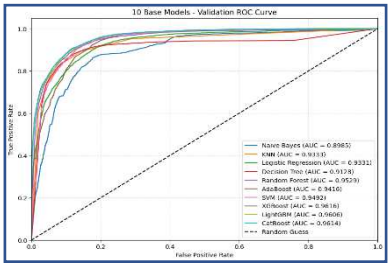

Fig 4. Validation ROC for 10 Base Model

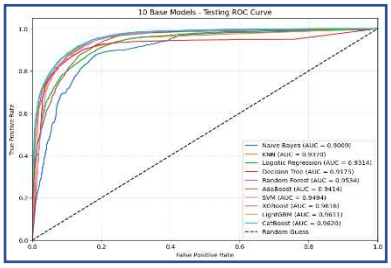

Fig 5. Testing ROC for 10 Base Model

Fig 3. – Fig 5. Shows that The ROC-AUC results between the ten different standard ML models are very different in distinguishing things in training, validation and testing dataset. In training, Decision Tree got the best AUC of 0.9807, followed by Random Forest with an AUC of 0.9791, CatBoost with an AUC of 0.9674, and XGBoost with an AUC of 0.9670. In contrast, Decision Tree had much lower AUC on validation (0.9128) and testing (0.9175) datasets which means that it was not that good at generalisation. The same reason also led to high AUC value for the training (0.9791) but lower AUC values for validation (0.9529) and testing (0.9534) of Random Forest.

In contrast, CatBoost and XGBoost had more stable results in all three datasets. The AUCs of CatBoost were 0.9674, 0.9614 and 0.9620 for training, validation and testing, respectively. Similarly, XGBoost obtained 0.9670, 0.9616 and 0.9616 on the same datasets. LightGBM also presented stable discrimination performance with AUC values of 0.9645, 0.9606 and 0.9611. The only classifiers that performed consistently well across all three datasets were SVM and AdaBoost. The AUC values for AdaBoost were 0.9417, 0.9416 and 0.9414 and for SVM were 0.9496, 0.9492 and 0.9494. The KNN achieved a small drop from 0.9554 on training to 0.9333 on validation and then increased to 0.9370 on testing. Logistic Regression showed relatively consistent AUC values of 0.9347, 0.9331, and 0.9314 . Naive Bayes had the lowest AUC values of all the classifiers with 0.8985 in training, 0.8985 in validation and 0.9009 in testing. CatBoost had the best AUC score overall at 0.9620 among all the classifiers and was followed closely by XGBoost (0.9616) and LightGBM (0.9611). The results show that the boosting-based models were more capable of discriminating items than most of the other individual classifiers.

2. Voting Ensemble Models: ROC-AUC Performance

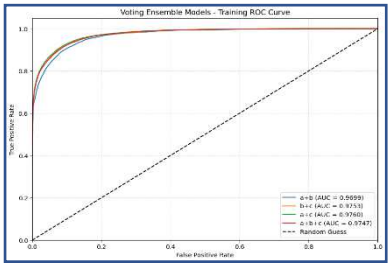

Fig 6. Training ROC for 4 ensemble Model

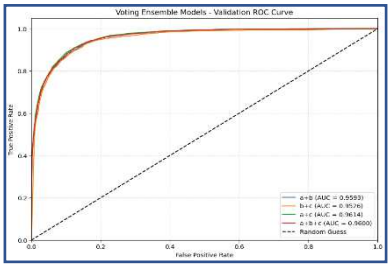

Fig 7. Validation ROC for 4 ensemble Model

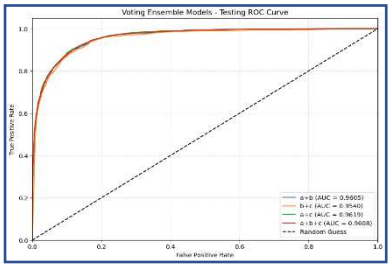

Fig 8. Testing ROC for 4 ensemble Model

Fig 6. – Fig 8 , We evaluated the voting ensemble models with training, validation and testing datasets to see how different combinations of classifiers performed at discriminating between things.
The a+c ensemble achieved the highest AUC of 0.9760 during training, followed by b+c (0.9753), a+b+c (0.9747) and a+b (0.9699). All four combinations did a very good job of classifying things in training.
Lastly, the combination of a+c achieved the best AUC score of 0.9614 on the validation dataset. The next group a+b+c had AUC of 0.9600. The groups a+b and b+c got 0.9593 and 0.9526 respectively. Based on these results, this a+c combination also had better discriminatory performance during validation.

The same tendency was noticed on the testing data set. The best AUC was obtained in a+c group, 0.9619, followed by a+b+c, 0.9608, a+b, 0.9605 and b+c, 0.9540. There are not many differences between the AUC values on validation and testing, meaning that the discriminatory performance is the same for both data sets.

It was the best overall performance of all four ensemble combinations with AUC values of 0.9760, 0.9614 and 0.9619 in training, validation and testing respectively. The combination of a+b+c also had competitive performance with AUC of 0.9747, 0.9600 and 0.9608. Generally, the results indicate that the a+c ensemble was the best discriminating between the voting ensemble combinations that were examined. Its testing AUC was still slightly less than that of CatBoost (0.9620), which was the best single classifier (0.9619).

The comparison ROC-AUC analysis indicates that both voting ensemble approaches and individual machine learning classifiers did a great job to tell the difference between phishing websites and real websites. The CatBoost classifier achieved the best test score of 0.9620 among all individual classifiers and the a+c voting ensemble achieved the best score of 0.9619 among all ensemble combinations. The ensemble approach performed pretty well but did not do better than the best individual classifier in terms of testing AUC. One important thing to note is that some models such as Decision Tree and Random Forest received very high AUC values when they were being trained but got large drops in performance when they were being validated and tested. CatBoost, XGBoost, LightGBM and the a+c ensemble, in contrast, had AUC values that were relatively stable across all three datasets. The results show that it is important to evaluate models on a variety of datasets, and not only on their performance during training. In terms of the overall ROC-AUC results, CatBoost showed the best discriminatory performance among the individual classifiers and a+c was the best voting ensemble combination.

4.5. Comparative Performance of Individual Machine Learning Models

To determine the effectiveness of ten machine learning classifiers (Naive Bayes, K-Nearest Neighbours (KNN), Logistic Regression, Decision Tree, Random Forest, AdaBoost, Support Vector Machine (SVM), XGBoost, LightGBM, and CatBoost) in phishing detection. The models were evaluated using accuracy, precision, recall, F1-score, Mean Squared Error (MSE) and Area Under Receiver Operating Characteristic Curve (AUC). The comparative results over the testing dataset are shown in Table 1.

Table 2. Performance Comparison of Individual Machine Learning Classifiers

Model	Accuracy	Precision	Recall	F1-Score	MSE	AUC
Naive Bayes	0.7108	0.8916	0.2323	0.3686	0.2892	0.9009
KNN	0.8862	0.8490	0.8352	0.8420	0.1138	0.9370
Logistic Regression	0.8646	0.8338	0.7836	0.8079	0.1354	0.9314
Decision Tree	0.8790	0.8193	0.8590	0.8387	0.1200	0.9175
Random Forest	0.8858	0.8180	0.8817	0.8487	0.1142	0.9534
AdaBoost	0.8666	0.8039	0.8368	0.8201	0.1334	0.9414
SVM	0.8796	0.7978	0.8955	0.8438	0.1204	0.9494
XGBoost	0.8856	0.8032	0.9072	0.8521	0.1144	0.9616
LightGBM	0.8834	0.7978	0.9094	0.8500	0.1166	0.9611
CatBoost	**0.8878**	0.8096	0.9037	**0.8541**	**0.1122**	**0.9620**

Table 2. indicate that the classifiers evaluated differed in their behaviour in the prediction task. CatBoost was the best to classify each item with accuracy 88.78%, KNN (88.62%) and Random Forest (88.58%) . The results indicate that both boosting-based and neighborhood-based learning techniques are able to model the features of phishing and real websites within the defined feature space.

The highest recall was achieved by LightGBM (90.94%), followed by XGBoost (90.72%) and CatBoost (90.37%). Higher recall is particularly important for finding phishing sites because it shows how many real phishing sites the model correctly identified. This made these three boosting-based classifiers pretty good at finding phishing attempts.

CatBoost also had the highest individual F1 score of 85.41% showing a good trade-off of accuracy and recall. XGBoost and LightGBm came in next with F1 scores of 85.21% and 85.00%. Furthermore, CatBoost attained the highest individual AUC of 96.20%, indicating its high ability to distinguish between real and fake cases at all classification thresholds. The Naive Bayes, however, got the lowest accuracy of 71.08% and F1-score of 36.66%. It had a fairly good accuracy rate of 89.16% but very poor recall rate of 23.23%. So, this means that the classifier identified only a small proportion of real phishing websites. Thus, the high accuracy did not lead to a satisfactory overall detection performance. The results of the comparison show that there was not one best classifier for all evaluation metrics . Overall, CatBoost performed the best on each task, while LightGBM and XGBoost performed better on recall. As the classifiers were not behaving the same, the researchers explored the use of ensemble learning to combine predictive traits that work well together.

4.6. Comparative Evaluation of Soft-Voting Ensemble Models

After evaluating individual classifiers, three robust and complementary models were selected for ensemble building: CatBoost (a), KNN (b) and Random Forest (c). The classifiers have been selected based on their individual classification performance and their suitability to combine different learning mechanisms.

CatBoost uses gradient boosting with ordered boosting, KNN uses neighbourhood similarity to detect patterns, and Random Forest uses bootstrap aggregation of multiple decision tree models. Their different learning mechanisms provide an opportunity to capture complementary patterns in the characteristics of phishing web sites.

Four soft-voting combinations were designed:

• a+b: CatBoost + KNN
• b+c: Random Forest + KNN
• a+c: Random Forest + CatBoost
• a+b+c: KNN + Random Forest + CatBoost

Soft voting takes the output probabilities of the classes from each classifier and averages them, and the highest probability class becomes the final prediction. In soft voting, we take the probabilistic output of each of the individual models, while in hard voting, we just take the hard predictions from each model.

Table 3. Performance Comparison of Soft-Voting Ensemble Models

Ensemble	Accuracy	Precision	Recall	F1-Score	MSE	AUC
a+b (CatBoost + KNN)	0.8918	0.8356	0.8740	0.8544	0.1082	0.9605
b+c (KNN + Random Forest)	0.8907	0.8415	0.8614	0.8513	0.1093	0.9540
a+c (CatBoost + Random Forest)	0.8912	0.8178	**0.9012**	**0.8575**	0.1088	**0.9619**
a+b+c (CatBoost + KNN + Random Forest)	**0.8922**	0.8341	0.8779	0.8555	**0.1078**	0.9608

The four ensembles combine different sets of classifiers to improve accuracy. Table 3 defines All of the four ensembles have sensitivity greater than 89%. The ensemble of three classifiers (a+b+c) has the highest accuracy (89.22%), which is greater than the highest accuracy achieved by an individual classifier (CatBoost, 88.78%). Hence, the accuracy gain achieved by using ensemble of three classifiers is approximately of (89.22% − 88.78%) = 0.44%.

In terms of MSE, the three models ensemble (a+b+c) achieved the lowest MSE (0.10778) among all the four ensembles. However, the ensemble a+c yielded the highest F1-score (85.75%) and recall (90.12%) among all ensembles. This means that the ensemble a+c detects a comparatively higher number of phishing websites as compared to the other ensembles,

elicited comparatively no of false-positives (slightly lower precision, 81.78%). Likewise, the adsorption ROC curve for the ensemble a+c has the highest AUC (96.19%), which is slightly higher than the AUC (96.08%) for the ensemble a+b+c. The ensemble a+b yielded the highest precision (83.56%) among all ensembles with the second-highest accuracy (89.18%). Ensemble a+b provides a trade-off between the detection of phishing websites and false-positives. It can be concluded that the ensemble of three classifiers (a+b+c) is better than any individual classifier and yields the best accuracy and MSE as compared to other ensembles. Nevertheless, it is outperformed by the ensemble a+c in respect of recall, F1-score, and AUC, which indicates that there is not a linear relationship between the number of classifiers and the performance of the ensemble.

4.7 Cross-Validation Performance Analysis

The standard machine learning algorithms were tested for their ability to predict, remain stable and work with other data using a method called cross-validation (CV). For each ensemble configuration and each classifier, the mean CV score and standard deviation were computed. The mean of the CV tells us how good the model is on average over those folds. The standard deviation of the CV tells us how different it's performance is across the validation folds. The higher mean of the CV indicates that we can make better predictions. A low standard deviation indicates a more stable and consistent model.

Table 4. Comparison of Cross Validation (Meaan & Standard Deviation)

Model	CV mean	CV Standard Deviation
Naive Bayes	0.620879	0.000868
KNN	0.882278	0.000978
Logistic Regression	0.846524	0.000966
Decision Tree	0.89149	0.001982
Random Forest	0.897421	0.002257
AdaBoost	0.859076	0.002459
SVM	0.881808	0.001725
XGBoost	0.894695	0.002573
LightGBM	0.892009	0.002169
CatBoost	0.895137	0.002162
a+b	0.891903	0.001774
b+c	0.892354	0.001862
a+c	0.895617	0.001809
a+b+c	**0.898582**	0.002104

Base Model Performance

According to Table 4, The worst performer in the machine learning classifiers was the Naive Bayes model with a CV mean of 0.620879 and a standard deviation of 0.000868. It was not great at prediction but it had the least amount of change across the validation folds. Its CV mean was 0.882278 with a standard deviation of 0.000978, meaning it was very good at predicting and very reliable. The CV mean for logistic regression is 0.846524 and the standard

deviation is 0.000966. This tells us that, on average, it does a decent job of classifying data and doesn't fluctuate much.

The Decision Tree classifier is able to make good predictions since its CV mean is 0.891490 and its standard deviation is 0.001982. The average cross validation score for Random Forest was in addition 0.897421 with a standard deviation of 0.002257. By this score this was the best individual classifier. This result implies that ensemble based tree learning is a good way of capturing complicated patterns in a data set.

The CV mean for Support Vector Machine (SVM) was 0.881808 and the standard deviation was 0.001725. The CV mean for AdaBoost was 0.859076 and the standard deviation was 0.002459. The results show that SVM outperformed AdaBoost in terms of average cross validation performance and the results were consistent .

We also tried three algorithms with gradient boosting: XGBoost, LightGBM and CatBoost. For XGBoost the CV mean was 0.894695 and the standard deviation was 0.002433. CatBoost did a little better. The CV mean was 0.895137 with a standard deviation of 0.002162 . LightGBM did better. The CV mean was 0.892009 with a standard deviation of 0.002169 . In cross-validation, CatBoost was the best performing model on average, with XGBoost and LightGBM coming in second and third place, respectively. LightGBM had a little less variance than XGBoost and CatBoost.

Ensemble Model Performance

To improve predictions even better, ensemble configurations were created by combining three individual classifiers: Random Forest (c), K-Nearest Neighbors (b) and CatBoost (a). We tried four ensemble combinations to test how well the combination of these classifiers works. CatBoost + KNN (a+b) together got a CV mean of 0.891903 and a standard deviation of 0.001774. The KNN + Random Forest (b+c) combination had a CV mean of 0.892354 and a standard deviation of 0.001862. The mean of the CV for the combination of CatBoost + Random Forest (a+c) was higher with 0.895617 and standard deviation was 0.001809. What these results show is that on average cross-validation, the combination CatBoost + Random Forest performed better than the other two-model ensemble configurations. The best result was obtained when all three classifiers were combined: CatBoost + KNN + Random Forest (a+b+c). It generated the highest CV mean of 0.898582 and the lowest standard deviation of 0.002104. This was the best average cross-validation score among all fourteen tested configurations, including single classifiers and ensembles. The result indicates that the complementary learning abilities of CatBoost, KNN and Random Forest can be combined to improve the overall prediction performance compared to using each one alone or with two models.

Overall Comparative Analysis

The CatBoost + KNN + Random Forest ensemble was the best overall cross-validation performance of fourteen configurations with a CV mean of 0.898582 (89.86%). The next were Random Forest (score 0.897421, 89.54%), CatBoost + Random Forest (score 0.895617, 89.56%), CatBoost (score 0.895137, 89.51%) and XGBoost (score 0.894695, 89.57%). The least variation in the data was observed with Naive Bayes (0.000868), followed by Logistic Regression (0.000966) and then KNN (0.000978). The proposed three model ensemble did not

have least standard deviation but its value was 0.002104 which showed that there was not much difference between the validation folds.

Overall, these cross-validation results indicate that the proposed CatBoost + KNN + Random Forest ensemble achieves the best average prediction performance among all configurations tested. Its mean CV of 0.898582 is about 0.12 percentage points higher than Random Forest, which was the best individual classifier (0.897421). “The combined ensemble can be considered as the best configuration by the cross-validation.

4.8 Explainable AI

Feature Selection by ANOVA Analysis

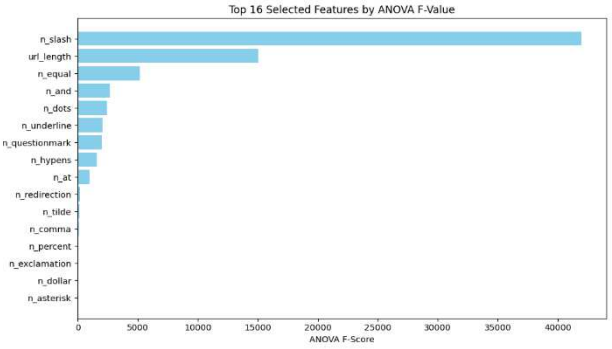

Fig. 9 Feature Selection by ANOVA Analysis

Before training the model, unnecessary or redundant noise in the benchmark dataset was removed and the number of dimensions of the features was reduced using one-way Analysis of Variance (ANOVA F-test). The ANOVA F-score distribution on Fig 9. shows that the feature selection protocol has successfully explored all the lexical and Structural attributes and kept only the top 16 most discriminative features that have the most statistical difference between real and fake URL classes. Specifically, structural attributes like {n_slash} and {url_length} received the highest F-scores by a considerable margin (over 41,000 and 15,000 respectively), which demonstrate that they are fundamental for distinguishing good from bad web structures. Then there are intermediate discriminative markers such as n_equal, n_and, n_dots, n_underline and n_questionmark. Symbols with little variation across target categories such as `n_space, `n_hastag, and `n_plus` were successfully filtered out.

Model Interpretability via SHAP Analysis

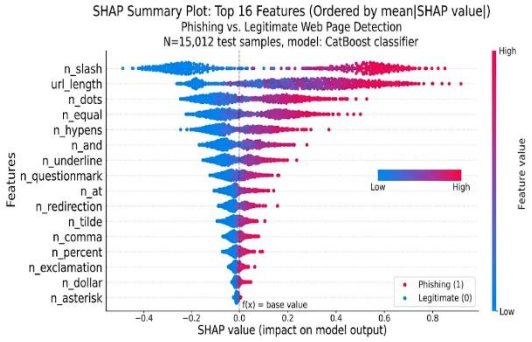

Fig. 10 SHAP Summary Plot: Top 16 Features

To learn how the proposed CatBoost classifier made its decisions, and to ensure that it was interpretable, we used SHapley Additive Explanations (SHAP) on the independent test set (n = 15,012 samples). The global SHAP summary (beeswarm) plot in Fig. 10 ranks the 16 input features from highest to lowest by their mean absolute SHAP value. Each point is an individual instance of a URL and the horizontal position represents the effect of the feature on the model output compared to the base value f(x). The color gradient represents the magnitude of the feature value from blue (low) to red/pink (high). Structural and lexical URL features are the most important features for the prediction performance of the classifier, as shown in Figure 10. In particular, n_slash appears to be the best predictor, closely followed by url_length, n_dots, n_equal and n_hyphens. Higher feature values (in red and pink) consistently yield positive SHAP values, reaching up to +0.8 for these top-ranked attributes, which heavily skews the model's prediction toward the phishing class (1). Conversely, URLs with fewer special characters, fewer slashes, and shorter lengths (blue) tend to be on the negative side of the baseline, shifting the output toward the legitimate class (0). At higher frequencies secondary features such as n_and, n_underline, n_questionmark, n_at and n_redirection contribute small but positive contributions when they are present. Conversely, the low-impact lexical symbols at the bottom of the plot (n_tilde, n_comma, n_percent, n_exclamation, n_dollar, and n_asterisk) are close to the zero-impact line (0.0) meaning they do not have much effect on the overall classification boundary.

The CatBoost classifier that we proposed was evaluated for transparency and further insight into the decision-making process using SHapley Additive Explanations (SHAP) on the independent test set (n= 15,012 samples). The global SHAP summary (beeswarm) plot in Fig. 10 ranks the 16 input features from highest to lowest by their mean absolute SHAP value. Each point is an individual instance of a URL and the horizontal position represents the effect of the feature on the model output compared to the base value f(x). The color gradient represents the magnitude of the feature value from blue (low) to red/pink (high). Structural and lexical URL features are the most important features for the prediction performance of the classifier, as shown in Figure 10. In particular, n_slash appears to be the best predictor, closely followed by url_length, n_dots, n_equal and n_hyphens. Higher feature values (in red and pink) consistently yield positive SHAP values, reaching up to +0.8 for these top-ranked attributes, which heavily skews the model's prediction toward the phishing class (1). Conversely, URLs with fewer special characters, fewer slashes, and shorter lengths (blue) tend to be on the negative side of the baseline, shifting the output toward the legitimate class (0). At higher frequencies secondary features such as n_and, n_underline, n_questionmark, n_at and n_redirection contribute small but positive contributions when they are present. Conversely, the low impact lexical symbols at the bottom of the plot, n_tilde, n_comma, n_percent, n_exclamation, n_dollar, n_asterisk, remain near the zero impact line (0.0) showing that they do not have much effect on the overall classification boundary.

Local Interpretability via LIME Analysis

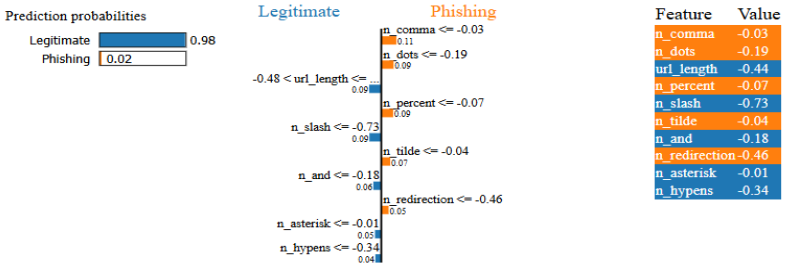

Fig 11. Local Interpretability via LIME Analysis

On Figure 11 , Global feature importance shows general trends, and Local Interpretable Model-agnostic Explanations (LIME) was added to provide instance-level details on how each URL is classified. The LIME local explanation result shows that the framework can clearly explain the decision-making process of a typical test case. The classifier correctly predicted the URL as Legitimate with a high confidence of 0.98 (versus a very low opportunity of 0.02 for phishing). The local feature weight contributions calculated from normalized attribute thresholds like `n_comma`, `n_dots`, `url_length`, `n_percent` and `n_slash` clearly show how local decision rules ensure the instance under inspection to be safe and build trust on the model's predictions.

4.9. Reliability and Probabilistic Performance Evaluation

In addition to the classification accuracy, the reliability of evaluated models was assessed by means of Matthews Correlation Coefficient (MCC) and Brier score. MCC considers all four entries of the confusion matrix and thus gives a balanced evaluation of classification quality. Its value ranges from -1 to +1, with +1 being perfect classification, 0 being performance no better than random classification, and -1 being complete disagreement.
The Brier score is a measure of the mean squared error between predicted probabilities and actual class outcomes. Lower values are better, in terms of probabilistic prediction.

Table 5. Reliability Evaluation of Individual and Ensemble Models

Model	MCC	Brier Score
Naive Bayes	0.355192	0.287578
KNN	0.758507	0.082045
Logistic Regression	0.704396	0.102345
Decision Tree	0.747495	0.089543
Random Forest	0.762667	0.078597
AdaBoost	0.714464	0.156211
SVM	0.749639	0.086518
XGBoost	0.763059	0.077595
LightGBM	0.759561	0.078508
CatBoost	0.766474	0.077183
a+b	**0.772698**	0.075064
b+c	0.764666	0.076727
a+c	0.770313	0.076366
a+b+c	0.771534	**0.074978**

Table 5. indicate that CatBoost achieved highest MCC among the individual classifiers with a value of 0.766474 . followed by Random Forest and XGBoost with 0.762667 and 0.763059 respectively . These values suggest that there is a large degree of agreement between the actual and predicted classifications.
The highest MCC of 0.772698 was achieved by the ensemble combinations of a + b. The proposed a+b+c combination had a slightly worse performance with an MCC of 0.771534. The difference is small but it shows that adding Random Forest to the CatBoost-KNN combination did not improve the MCC.

However, the three-model ensemble has the minimum Brier score (0.074978), indicating the lowest predicted probabilistic error among the tested models. The a+b combination also produced a Brier score of 0.075064 which was very competitive.

Naive Bayes was the least reliable with a Brier score of 0.287578 and MCC of 0.355192. This is in accordance with its high classification error and low recall.

Finally, the reliability evaluation shows that the proposed ensemble provides a competitive combination of quality in terms of probability prediction and classification agreement. However, the a+b combination is the best model in terms of MCC, while the a+b+c combination is the best model in terms of the lowest Brier score.

4.10. Computational Efficiency and Operational Feasibility

We evaluated operational feasibility of the evaluated models with four metrics: in-memory size, inference latency per sample, training time and serialised disc size. These indicators are an indication of the computational resources needed to store, predict and train the model.

Table 6. Operational and Computational Evaluation

Model	Training Time (s)	Latency (ms/sample)	Memory (bytes)	Disk Size (bytes)
Naive Bayes	0.114858	0.001047	48	1,096
KNN	0.008559	0.401173	48	12,131,892
Logistic Regression	0.304367	0.000039	48	844
Decision Tree	0.281868	0.000177	48	660,926
Random Forest	4.853409	0.010939	48	48,734,591
AdaBoost	1.254291	0.001643	48	28,951
SVM	765.557800	1.141338	48	39,481,520
XGBoost	0.462234	0.001471	48	299,084

LightGBM	0.688329	0.003615	48	353,580
CatBoost	15.640250	0.000941	48	1,119,686
a+b	16.265690	0.327572	48	26,502,790
b+c	4.136975	0.342841	48	121,731,598
a+c	20.333210	0.011767	48	99,707,582
a+b+c	19.741600	0.386299	48	123,970,695

Training Time: The amount of time required to train a model on the training data is known as training time. According to Table 6. The individual classifier that took the least time to train was KNN, at approximately 0.0086 seconds, while SVM took approximately 765.56 seconds. This big difference is a result of the computational properties of the respective learning algorithms. The least training time was needed for ensemble combinations b+c and a+c, approximately 4.14 and 20.33 seconds respectively. The proposed a+b+c ensemble took an extra 19.74 seconds to train compared to the fastest individual and ensemble alternatives.

Inference Latency: Inference latency is the time it takes to generate a prediction for a given sample. The lowest reported latency was 0.000039 ms/sample for Logistic Regression, while SVM took about 1.141338 ms. The proposed ensemble of three models got a latency of 0.386299 ms per sample. We expect the ensemble to have a higher inference latency than multiple individual classifiers as the ensemble needs to generate and combine probability predictions from multiple constituent models. However, the reported latency provides a baseline to assess its suitability for time-critical detection applications.

Memory: The memory footprint The table shows the in-memory size of each evaluated model, which was 48 bytes. This value is not showing any significant differences in the real memory requirements of the model, as it is constant for all classifiers. Therefore, additional validation of the memory measurement method is needed before any conclusions about the efficiency of memory can be made.

Serialized Model Size: Serialized disc size is the size of the storage required to save a trained model. Random Forest required around 48.73 MB while Logistic Regression required only 844 bytes . Among the combinations considered, the proposed a+b+c ensemble was found to have the maximum storage requirement with an estimated 123.97 MB. These results reveal an important trade-off between computation requirements and classification performance. Although the proposed ensemble achieved the highest accuracy, the training time and the storage requirements were higher than several individual classifiers. Therefore, the deployment decisions should consider the importance of detection accuracy, the availability of hardware resources and the application requirements.

Discussion

The combined experimental results provide several important observations regarding the effectiveness of individual classifiers and ensemble learning for phishing detection.First, CatBoost demonstrated the strongest individual classification performance, achieving 88.78%

accuracy, 85.41% F1-score, and 96.20% AUC. Its performance established a strong baseline for constructing ensemble combinations.

Second, the soft-voting evaluation demonstrated that combining heterogeneous classifiers can improve classification accuracy. The proposed CatBoost-KNN-Random Forest ensemble achieved 89.22% accuracy, representing an improvement of approximately 0.44 percentage points over the best individual classifier. It also obtained the lowest MSE and Brier score among the evaluated ensemble combinations.

Third, the comparative results revealed that different ensemble configurations provide different advantages. The CatBoost-Random Forest combination achieved the highest recall, F1-score, and ensemble testing AUC, whereas the CatBoost-KNN combination achieved the highest MCC. Therefore, the best ensemble depends on the performance criterion considered.

Fourth, ROC and confusion matrix analyses provided additional evidence of the models' classification behavior. The proposed ensemble maintained a testing AUC of 96.08% and correctly identified 4,788 phishing instances. However, its 666 false-negative cases indicate that further improvements in phishing detection sensitivity are possible.

Comparative Analysis with Existing Studies

To evaluate the effectiveness and contributions of the proposed phishing detection framework, a comparative analysis was conducted with recent studies on machine learning, ensemble learning, feature selection, and explainable phishing detection. Table X summarizes their methodologies, major contributions, limitations, and how the proposed framework extends existing approaches.

Table 7. Comparative analysis of existing phishing detection studies and the proposed framework

Study	Main Approach	Best Result / Contribution	Limitation	Contribution of Proposed Study
Mushtaq et al. [9], 2024	Ensemble learning for URL phishing detection.	Evaluated feature selection and ensemble-based detection.	Limited comparison with the broader evaluation framework used here.	Evaluates 10 classifiers and four voting ensembles with predictive and operational metrics.
Lightweight Explainable Detection [10], 2025	RFE, SMOTE, XGBoost and SHAP.	85.4% accuracy using four selected features.	Limited feature representation and F1-score of 0.558.	Achieves 89.22% accuracy using ANOVA-selected 16 features and a three-model ensemble.

Rashid et al. [11], 2024	Unsupervised domain adaptation for cross-dataset URL detection.	Improved cross-dataset F1-score by 0.06 on average.	Focuses primarily on domain adaptation.	Adds MCC, MSE, Brier score, latency, memory and model-size evaluation.
Ghalechyan et al. [12], 2024	Deterministic and probabilistic neural networks.	Investigated neural-network detection using public and production URL datasets.	Limited emphasis on the comparative soft-voting and operational framework used here.	Compares 10 classifiers and four voting ensembles using multiple performance metrics.
Jain et al. [13], 2026	Boruta feature selection, Optuna optimization and stacking.	Achieved 96.15% accuracy and 96.56% F1-score.	Uses a different stacking and optimization strategy.	Evaluates ANOVA-based feature selection and multiple soft-voting combinations.
Khan et al. [14], 2026	Bayesian-optimized stacking with SHAP-guided feature learning.	Integrates optimization, ensemble learning and explainability.	Different ensemble architecture and evaluation focus.	Combines SHAP and LIME with predictive and computational evaluation.
Dandotiya et al. [15], 2026	Machine learning-based browser extension.	Focuses on real-time phishing detection using URL and visual information.	Primarily focuses on browser-based deployment.	Adds comparative model evaluation and probability-based reliability metrics.
Panthi et al. [16], 2026	SHAP and LIME for	Evaluates interpretability of GBM, XGBoost and LightGBM.	Primarily focuses on explainability of boosting models.	Extends evaluation to 10 classifiers and four ensembles with operational metrics.

	interpreting boosting models.			
Prasad and Chandra [17], 2024	PhiUSIIL framework using similarity index and incremental learning.	Introduced a large-scale dataset with 235,795 URLs.	Focuses on similarity-based detection and incremental learning.	Investigates ANOVA-based feature selection, SMOTE and voting-based classification.
Aljofey et al. [18], 2022	Phishing detection using URL and HTML features.	Developed an effective feature-based phishing website detection approach.	Different feature and evaluation framework.	Provides a broader comparative evaluation with ensemble learning and explainability.
Proposed Study (2026)	**ANOVA top-16 features, SMOTE, 10 classifiers, five-fold CV and four soft-voting ensembles.**	**a+b+c (CatBoost + KNN + Random Forest): 89.22% accuracy, 85.55% F1-score, 96.08% AUC, 0.7715 MCC and 0.07498 Brier score.**	**Requires further external cross-dataset and real-world deployment validation.**	**Integrates predictive performance, feature selection, reliability, computational efficiency and SHAP–LIME explainability in one framework.**

After Evaluating Table 7. The comparative analysis reveals that previous works have studied phishing detection using feature selection, machine learning, neural networks, domain adaptation, ensemble learning, and explainable artificial intelligences. However, the proposed framework integrates these evaluation dimensions by systematically comparing ten individual classifiers and four voting ensemble configurations. a+b+c (CatBoost + KNN + Random Forest) 89.22 % 85.55 % 0.9608 0.7715 0.07498 Unlike approaches that focus on classification accuracy or a specific detection technique, the proposed framework takes into account model interpretability, computational efficiency, and probability reliability. Hence, its main

contribution is to integrate multiple evaluation dimensions into a unified phishing detection framework, but not to claim better accuracy than any existing study.

To summarise, the operational evaluation indicated that ensemble-based detection requires extra storage and training resources. These results suggest that predictive performance should be considered along with computational efficiency when choosing models for practical deployment.

To summarise, the experimental results confirm the effectiveness of the proposed soft-voting framework for combining Random Forest, KNN and CatBoost to detect phishing websites. The main advantage is the ability to preserve competitive F1-score, AUC, MCC and probabilistic reliability, while achieving the highest accuracy among the evaluated models. The feature selection, class balancing, operational evaluation and multiple performance indicators are combined to provide a solid experimental basis for the proposed phishing detection approach.

6. Conclusion and Future Work

In this research, we presented a complete framework for detecting phishing websites with machine learning. The framework involves feature selection, class balancing, single classification, and ensemble learning methods. The proposed framework identified the top 16 relevant features using ANOVA-based feature selection and then used SMOTE to handle the class imbalance. The ten individual machine learning classifiers and four soft-voting ensemble combinations were assessed using separate training, validation, and testing datasets as well as five-fold cross-validation. The CatBoost-KNN-Random Forest (a+b+c) ensemble method proposed in this work achieved the best overall classification accuracy of 89.22% among the tested methods, with a precision of 83.41%, recall of 87.79%, and F1-score of 85.55%. It also achieved an AUC of 0.9608, an MCC of 0.7715, an MSE of 0.10778 and a Brier score of 0.07498. The results demonstrate that an efficient detection of phishing can be achieved by using soft voting to combine complementary classification algorithms. The model interpretability is also improved by using SHAP and LIME together to provide global and local explanations of prediction results. Besides predictive performance, the evaluation of inference latency, memory consumption, training time and serialized model size helps to elucidate the computational requirements of the proposed framework. Overall, the proposed method offers a complete and understandable solution to the detection of phishing websites by combining computational efficiency, prediction reliability and classification effectiveness in a unified framework.

The experimental results are promising and there are still many chances for further improvement. Future work will assess the generalization capability of the proposed framework using cross-dataset evaluation on independently collected datasets of legitimate URLs and phishing URLs. Secondly, it can be explored more sophisticated feature selection and optimization techniques such as automated hyperparameter optimization and evolutionary algorithms to enhance the classification performance and decrease the computational complexity. Third, the proposed ensemble can be extended by incorporating hybrid architectures and deep learning to capture more complex phishing patterns. Fourth, the framework can be investigated under practical deployment conditions in future work using browser extensions or web-based detection systems for real-time phishing detection. Moreover, the model could be able to adapt to new phishing techniques through the integration of incremental learning approaches and continually updated threat intelligence. Further research on probability calibration and explainability techniques may improve the reliability of

predictions and the classification of suspicious websites more transparently. These recommendations will help to move on the proposed framework toward a more adaptive, practical and robust system of phishing detection.

References (IEEE Style)

[1] A. Safi and S. Singh, “A systematic literature review on phishing website detection techniques,” Journal of King Saud University–Computer and Information Sciences, vol. 35, no. 2, pp. 590–611, 2023. doi: 10.1016/j.jksuci.2023.01.004

[2] Anti-Phishing Working Group (APWG), “Phishing Activity Trends Report, 4th Quarter 2025,” 2026. Available: APWG Phishing Activity Trends Report

[3] E. Kytidou et al., “Machine learning techniques for phishing detection: A review of methods, challenges, and future directions,” Journal of Computational and Theoretical Nanoscience's associated publisher platform — Sage Journals, 2025. doi: 10.1177/18724981251366763

[4] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from URLs,” Expert Systems with Applications, vol. 117, pp. 345–357, 2019.

doi: 10.1016/j.eswa.2018.09.029

[5] “Feature Engineering for Phishing Website Detection Using Machine Learning: A Systematic Review,” IEEE Access, vol. 13, pp. 192080–192104, 2025.doi: 10.1109/ACCESS.2025.3630334

[6] T. Kehkashan et al., “Explainable phishing website detection for secure and sustainable cyber infrastructure,” Scientific Reports, vol. 15, Art. no. 41751, 2025. doi: 10.1038/s41598-025-27984-w

[7] A. Safi and S. Singh, “A systematic literature review on phishing website detection techniques,” *Journal of King Saud University—Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, 2023, doi: 10.1016/j.jksuci.2023.01.004.

[8] “A state-of-the-art review on phishing website detection techniques,” *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3514972.

[9] S. Mushtaq, T. Javed, and M. Mohd Su’ud, “Ensemble learning-powered URL phishing detection: A performance driven approach,” *Journal of Informatics and Web Engineering*, vol. 3, no. 2, pp. 134–145, 2024, doi: 10.33093/jiwe.2024.3.2.10.

[10] “Lightweight and explainable phishing URL detection using feature selection and XGBoost,” in *Proc. 2025 International Conference on Intelligent & Innovative Practices in Engineering & Management*, 2025, doi: 10.1109/IIPEM65914.2025.11548404.

[11] F. Rashid, B. Doyle, S. C. Han, and S. Seneviratne, “Phishing URL detection generalisation using unsupervised domain adaptation,” *Computer Networks*, vol. 245, Art. no. 110398, 2024, doi: 10.1016/j.comnet.2024.110398.

[12] H. Ghalechyan *et al*., “Phishing URL detection with neural networks: An empirical study,” *Scientific Reports*, vol. 14, Art. no. 25134, 2024, doi: 10.1038/s41598-024-74725-6.

[13] A. Jain *et al*., “PhishNet 1.0: Optuna-optimized stacking ensemble with Boruta-based feature selection for phishing URL detection,” *Scientific Reports*, vol. 16, Art. no. 1480, 2026, doi: 10.1038/s41598-025-31447-7.

[14] H. A. Khan *et al*., “Explainable AI for phishing URL detection: A Bayesian-optimized stacking ensemble framework with SHAP-guided feature learning,” *Frontiers in Artificial Intelligence*, vol. 9, Art. no. 1854934, 2026, doi: 10.3389/frai.2026.1854934.

[15] M. Dandotiya *et al*., “Real time identification of phishing attacks through machine learning enhanced browser extensions,” *Scientific Reports*, 2026, doi: 10.1038/s41598-026-35655-7.

[16] S. Panthi *et al*., “A performance analysis of SHAP and LIME for interpreting boosting models in phishing URL detection,” *Scientific Reports*, 2026, doi: 10.1038/s41598-026-66800-x.

[17] A. Prasad and S. Chandra, “PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning,” *Computers & Security*, vol. 136, Art. no. 103545, 2024, doi: 10.1016/j.cose.2023.103545.

[18] A. Aljofey *et al*., “An effective detection approach for phishing websites using URL and HTML features,” *Scientific Reports*, vol. 12, Art. no. 8842, 2022, doi: 10.1038/s41598-022-10841-5.

[19] A. Hannousse and S. Yahiouche, “Towards benchmark datasets for machine learning based website phishing detection: An experimental study,” *Engineering Applications of Artificial Intelligence*, vol. 104, Art. no. 104347, 2021, doi: 10.1016/j.engappai.2021.104347.

[20] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, “Machine learning based phishing detection from URLs,” *Expert Systems with Applications*, vol. 117, pp. 345–357, 2019, doi: 10.1016/j.eswa.2018.09.029.

[21] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, “CANTINA+: A feature-rich machine learning framework for detecting phishing web sites,” *ACM Transactions on Information and System Security*, vol. 14, no. 2, Art. no. 21, 2011, doi: 10.1145/2019599.2019606.

[22] S. Marchal, J. François, R. State, and T. Engel, “PhishStorm: Detecting phishing with streaming analytics,” *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014, doi: 10.1109/TNSM.2014.2377295.

[23] A. Awasthi and N. Goel, “Phishing website prediction using base and ensemble classifier techniques with cross-validation,” *Cybersecurity*, vol. 5, Art. no. 22, 2022, doi: 10.1186/s42400-022-00126-9.

[24] A. Ezzat, "Web-page-phishing," *Kaggle*, [Online]. Available: https://www.kaggle.com/code/ahmedezzatibrahem/web-page-phishing/input. [Accessed: Nov. 14, 2026].

Author Contributions

Md. Sharfuddin and Md. Shakil contributed to the conception and design of the study, data acquisition, analysis and interpretation of the results. Both authors participated in drafting and critically revising the manuscript for important intellectual content. Both authors have read and approved the final version of the manuscript for submission and agree to be accountable for all aspects of the work.

Corresponding Author

Md. Sharfuddin is the corresponding author and guarantor of this manuscript. He will take responsibility for the integrity of the work as a whole, communicate with the journal throughout the submission and peer-review process, address reviewers' and editors' comments, and coordinate any necessary revisions on behalf of all authors.

Corresponding Author:
Md. Sharfuddin
Department of Computer Science and Engineering,
Southeast University, Dhaka, Bangladesh.
Email: sharfuddin6161@gmail.com

Acknowledgments

The authors would like to acknowledge the Department of Computer Science and Engineering, Southeast University, Dhaka, Bangladesh, for providing academic support and research facilities. No other individual contributions requiring acknowledgment were made.

Conflict of Interest Statement

The authors declare that they have no financial or personal relationships with any individuals or organizations that could inappropriately influence or bias the work presented in this manuscript.

Declaration of Funding

The authors received no specific financial support or research funding from any public, commercial, or non-profit funding agency for this study.

Role of the Funding Source

There was no external funding for this study. Therefore, no funding organization was involved in the study design, data collection, analysis, interpretation, manuscript preparation, or the decision to submit the manuscript for publication.

Ethical Approval and Compliance

This study was conducted in accordance with applicable ethical standards and research integrity principles. If publicly available, anonymized datasets were used, no new human or animal experiments were conducted as part of this study. All data were used in accordance with their respective terms of access and use.

Publication Ethics Statement

The authors certify that this manuscript complies with the journal's ethical principles for publication, including the Committee on Publication Ethics (COPE) guidelines. The authors confirm that the manuscript is original and free from fabrication, falsification, plagiarism, and other forms of research misconduct. The manuscript has not been published previously and is not under consideration for publication elsewhere.

Writing Assistance

The authors declare that no external professional writing assistance was received in the preparation of this manuscript.

Author Information

Author	Affiliation	Email
Md. Sharfuddin*	Southeast University	sharfuddin.seu@gmail.com
Md. Shakil	Southeast University	mdshakil3214@gmail.com

Corresponding author: Md. Sharfuddin.