

Trustworthy Phishing URL Detection Using Machine Learning: Performance Analysis and Future Directions

Omar Almomani^{1,*}, Ahmad Shalaldeh², Mohammed Amin Almaiah³, Ahmad Saleh Ali Shatat⁴, Mohammed Madi⁵, Mutasem K. Alsmadi⁶ and Ibrahim Mohammed Dweib⁷

¹ Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Al Salat 19111, Jordan.

² Department of Data Science and Artificial Intelligence, Al-Ahliyya Amman University, Al Salat 19111, Jordan.

³ King Abdullah the II IT School, University of Jordan, Amman, 11942, Jordan.

⁴ Department of Management Information System, College of Administrative Sciences, Applied Science University, Kingdom of Bahrain.

⁵ Networks and Database, Faculty of Computing and IT, Sohar University, Sohar, 311, Oman.

⁶ Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam 31441, Saudi Arabia.

⁷ Math & IT Department, Sultan Qaboos University, P.O. Box 43, Al Khoudh PC 123, Sultanate of Oman.

* Corresponding author: Omar Almomani. o.almomani@ammanu.edu.jo

Received: 15 April 2026 **Accepted:** 1 October 2026 **Published:** 20 October 2026

DOI: 10.15849/ijasca.129 **License:** Open access under CC BY 4.0. Authors retain copyright.

Publisher: Al-Zaytoonah University of Jordan (ZUJ), Amman, Jordan.

Cite this article: O. Almomani, A. Shalaldeh, M. Almaayah, A. S. A. Shatat, M. Madi, M. K. Alsmadi, and I. M. Dweib, "trustworthy phishing URL detection using machine learning: performance analysis and future directions," *Int. J. Advance Soft Compu. Appl.*, vol. 18, no. 3, pp. 532-560, 2026, doi: <https://doi.org/10.15849/ijasca.129>.

ABSTRACT

Over the years, Uniform Resource Locator (URL) Phishing has been a threat to online security because their scams exploit trust in a misleading URL to trick unsuspecting users into disclosing their username or passwords and other personal information. Not only that, but it is also crucial to identify these malicious URLs and ensure online platforms and users are protected. The ISCXURL2016 data was utilized for the purpose of analyzing the performance of different Machine Learning (ML) models in differentiating phishing and legitimate URLs in this paper. A wide range of classifiers were explored including the classic classifiers as well as more advanced ensemble classifiers. The results of the experiments indicate that ensemble-based boosting models can be more beneficial for URL phishing detection. HistGradientBoost achieved the highest accuracy of 98.47%, followed closely by LightGBM and XGBoost that both outperformed the accuracy of 98%. The other classifiers, RandomForest and ExtraTrees, also performed well with accuracy above 97% and traditional classifiers such as Quadratic Discriminant Analysis (QDA) showed the poorest performance with an accuracy of 67.75%. The results show that the boosting-based ensembles provide a suitable negation of the complex and non-linear structure of phishing URLs. Along with empirical evaluation, it offers a Properties & Positioning viewpoint as well as the importance of the ML-based phishing detection in the broader context of cybersecurity issues. Moreover, it discusses the future directions, with Federated Learning (FL) in the context of decentralized and privacy-preserving collaboration and Large Language Models (LLM) in the context of semantic reasoning and flexibility. These insights together will provide the building blocks for more intelligent, adaptive and trustworthy phishing organizations.

Keywords: Phishing URL; ISCXURL2016; Cybersecurity; Machine learning; Web Security.

1 Introduction

Due to the evanescent advancements of the internet and digital technology, there have been new kinds of interactions among people, corporations, and governments in transactions. Rapid digitization, however, poses great challenges to cybersecurity today [1-3]; URL phishing, spamming, defacement, and malware have become one of the most harmful and prevalent threats [4-6]. Phishing, which is largely through malicious URLs, consists of tricks to extract sensitive information from users, such as usernames, passwords, and financial and personal details [7-9]. Such attacks may also result in harsh effects such as financial loss, identity theft, and compromised security at the organizational level. As sophistication in URL phishing becomes greater, the blacklisting and heuristic detection methods of the past fail to prove

adequate [10, 11]. Thus, there is an urgent need for much more advanced and adaptive solutions for combating this ever-evolving threat .

URL phishing is an active, evolving threat and, therefore, a significant threat to the conventional methods of detection . Their previous blacklist-based blocks or ad hoc-based rules are lagging quite a bit behind the current tactics of the competitors' . The blacklists are implemented on bad URL lists that attackers can easily bypass by making simple modifications to the web addresses. Likewise, heuristic techniques are only sensitive to the well-known effects of malicious behavior and thus would not work against a zero-day attack or a novel evasion technique . Therefore, smart ideas to develop the already discovered trends further and evolve along with the ever-changing world of phishing attacks are demanded [12].

ML is at the forefront of the battle in the fight against URL phishing attacks. With ML, one can analyze large volumes of data and identify complex patterns that may indicate malicious conduct [13, 14]. With the help of ML algorithms, it becomes possible to create predictive models that automatically [15] detect and classify phishing URLs with very high accuracy [16-19]. Such a model becomes beneficial in real time, and early detection with damage limitation can be applied. However, success in these cases is largely dependent on the classifier used, dataset quality, and features used in training. Among the various classifiers available in ML [20] Ensemble methods such as XGBoost, AdaBoost, Gradient Boost, and CatBoost have shown their ability to deploy multiple weak learners to build a powerful predictive model. Similarly, traditional classifiers like Support Vector Machines (SVM), Random Forest, Decision Trees, K-Nearest Neighbors (KNN), as well as modern methodologies like Stochastic Gradient Descent (SGD), Light Gradient Boosting Machine (LightGBM), and Artificial Neural Networks (ANN), have gained wide popularity in cybersecurity applications [21-24].

There is a remarkable gap in the literature concerning ML in URL phishing [25, 26]. but it has great potential. There are several algorithms that have been tested in the existing literature, but they have been used in a variety of different experimental settings and on different data sets. This non-standardization makes the notion of any meaningful conclusions about the most suitable classifiers to detect phishing URLs very complicated. This makes it difficult for organizations to know what model to adopt to put into practice when they want to use an ML-based defense. To address this, the paper performs a comprehensive comparative study of 18 ML classifiers on ISCX-URL-2016 data and thus provides a uniform and repeatable evaluation process. The classifiers used were: XGBoost, Random Forest, Bagging, AdaBoost, Gradient Boosting, CatBoost, LightGBM, Logistic Regression, ExtraTrees, SVM, Decision Tree, SGD, KNN, Passive Aggressive, Ridge Classifier, QDA, LDA, and HistGradientBoost [27]. All the algorithms represent various types of learning paradigms, from ensemble/tree-based algorithms to linear/instance-based algorithms and provide a broad understanding of the state of the art in phishing detection. The performance of each classifier will be assessed based on the different metrics used to evaluate detection accuracy, detection precision, detection recall, F1-measure, and detection efficiency and the most appropriate ML classifiers for phishing URL detection will be recommended.

In contrast to previous studies that compared a handful of models or used a non-uniform validation process, the current study introduces a single framework of a 10-Fold Cross-Validation process for measuring a wide range of 18 machine learning classifiers on the ISCX-URL2016 dataset. By using this standardized procedure, the measurement of the performance of each model is fair, reliable and statistically reliable. This standardized experimental procedure guarantees equity, reliability, and statistical dependability in the evaluation of the performance of every model. Moreover, the analysis includes a more advanced preprocessing

and normalization pipeline, which constitutes the synthesis of the categorical encoding of the data and Min-Max scaling to ensure equal representation of the data and to address bias among features. In addition to the evaluation, the hybrid ensemble improvement framework is also suggested in this research, which conceptually joins the most successful boosting algorithms, namely HistGradientBoost, LightGBM, and XGBoost, with new Federated Learning (FL) and Large Language Model (LLM)-based detection algorithms. This hybrid vision forms the base of the decentralized, intelligent, and adaptive phishing detection that can be developed based on the current web threats. Taken together, these factors establish the novelty of the present work, reaching a step further than the traditional benchmarking and reaching a reproducible, extensible, and prospective machine learning pipeline to detect robust phishing URLs.

The main contributions of this paper are the following:

1. **Thorough Benchmarking:** a methodical comparison of the 18 ML classifiers on the ISCX-URL-2016 dataset to identify which sides of the classifier are strong and which are weak.
2. **Feature-Driven Insights:** Research into the effect of preprocessing (dealing with missing values, coding, and normalizing of categorical variables) on model results.
3. **Systematized measurement** in terms of accuracy, precision, F1-score, and recall, which are equal and comparable among classifiers.
4. **Solid Recommendations:** Quantitative advice on the best ML models to implement in practice to identify URL phishing.

The paper's organization is as follows: Section 2 contains the literature review on URL phishing detection using ML classifiers. Section 3 summarizes the methodology. A presentation of the experimental results with the implications on URL phishing detection would be contained in Section 4. The Trade-offs, Threats, and Roadmap discussion is in section 5. Finally, the work ends with Section 6, which offers a conclusion as well as suggestions for future studies.

2 Related Works

The section deals with the existing literature based on an ML classifier for URL phishing detection.

A study by O. K. Sahingoz, E. Buber, O. Demir, and B. J. E. S. w. A. Diri, [28]. In this study, a real-time anti-phishing system is proposed based on ML algorithms for the detection of phishing websites based on URL characteristics. Various classification algorithms are applied, including seven algorithms: Random Forest, Naive Bayes, KNN, Adaboost, K-Star, SMO, and Decision Trees with added NLP techniques and word vector-based features. The result is the creation of a large, balanced dataset of 73,575 URLs (36,400 legitimate and 37,175 phishing), forming the basis of extensive testing. The experimental results indicate Random Forest algorithm with only NLP-based features gives the best performance with a 97.98% accuracy rate for the detection of phishing URLs.

Another study by S. Shikalgar, S. Sawarkar, and S. J. I. J. E. R. T. Narwane, [29], This study proposes an ML-based approach to detect phishing websites by analyzing URL features. The study emphasizes the use of hybrid classification techniques to improve detection accuracy. The proposed phishing model relies on a dataset sourced from PhishTank, consisting of 2,905 URLs, both phishing and legitimate. The phishing model under consideration adopts a hybrid approach that employs the classifiers Naive Bayes, Random Forest, SVM, XGBoost, and Stacking. Experimental results indicate that SVM and stacking classifiers achieved the best

detection accuracy of 86.3% and 85.6%, respectively, thus showing superiority over Naive Bayes (80.2%) and XGBoost (85.5%).

In parallel, the study conducted by N. Nagy et al. [30], is a comprehensive objective of this study is to study the effectiveness of parallel algorithm for phishing URL detection using ML/DL models. The authors have been introduced to the sequential ML constraints and faced with large datasets and real-time detection, they put one foot in sequential and one in parallel to test it and assess its performance and computational efficiency. Four models were used: Random Forest (RF), Naïve Bayes (NB), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). The dataset has been generated using the malicious and benign Webpages dataset, consisting of 54000 training and 12000 testing samples with features extracted from URL content, network attributes and lexical properties. The results showed that the NB model was able to reach maximum accuracy of 96.01%, while RF, CNN and LSTM models were able to obtain a high recall value of 100%, which means that they have a high sensitivity to phishing URLs.

A study by S. Jalil, M. Usman, A. J. J. O., and A. I. Fong [21] This research is in the field of phishing detection using ML for categorizing phishing URLs without querying an external service or visiting the web pages. Authors propose a lexical feature-based ML methodology that extracts and analyzes features directly from the URL string. The proposed model is tested on six datasets with eight different ML classifiers, such as Naïve Bayes, logistic regression, SVM, K-nearest neighbor, Decision Tree (J48), AdaBoost, Multilayer Perceptron (MLP), and Random Forest. Interestingly, Random Forest got the best accuracy as high as 96.25% on Kaggle datasets. The proposed system surpasses the other existing methods in the field of phishing URL detection.

Another study by A. Karim et al. [31]. This work presents an ML approach for phishing website detection using URL features. The study uses over eleven thousand URLs and extracts thirty-three URL-based attributes to classify phishing and legitimate websites. The work used ML models like Decision Tree, Random Forest, Naive Bayes (NB), Gradient Boosting Machine (GBM), KNN, Support Vector Classifier (SVC), as well as a proposed hybrid LSD model, which comprises Logistic Regression + SVM + Decision Tree for classification by soft-voting and hard-voting. The canopy feature selection method, together with cross-fold validation and hyper-parameter tuning using grid search, was used to find the best features and increase model accuracy. Altogether, the LSD model proposed achieved maximum accuracy at 98.12%, which is much greater than any of the individual classifiers. Comparative analyses illustrate the best performance in tree-based ensemble models, while linear and probabilistic models were slightly inferior. This study demonstrated a sign of improvement in phishing detection accuracy via hybridization within the ML models.

R. Basnet, S. Mukkamala, and A. H. Sung, [32], came up with a study that identified 16 characteristics of phishing emails, including HTML format, the presence of IP-based URLs, domain age, number of links, presence of JavaScript, and keyword frequency, as the basis for distinguishing between phishing and legitimate emails. The database is divided into an even set of training and testing emails, with 973 phishing emails and 3,027 legitimate emails for each. Several machine learning algorithms such as SVM, neural network, self-organizing maps (SOMs), and K-means were implemented to obtain the best accuracy of 97.99% using SVM. The study has pointed out the success of feature based classification and recommended further studies related to feature selection and model optimization for better phishing detection accuracy.

Another study by Albishri and Dessouky [33] compared the various ML models in detecting URL phishing and highlighted the strength of the ensemble models, like Random Forest. They incorporated a dataset of more than 1.5 million entries (balanced on phishing and benign URLs) and combined powerful feature engineering methods (such as WHOIS data, use of HTTPS, JavaScript features, etc.) to improve the classification accuracy. With the help of the systematic hyperparameter optimization of the Random Forest model with the help of GridSearchCV, it was possible to obtain excellent accuracy of between 99.93 and 99.98, which outperformed traditional classifiers such as Naive Bayes and SVM. By performing the assessments during eight sequential days, the authors emphasized the significance of the model stability and generalization in real-life conditions. Their results showed that ensemble learning with feature optimization can reduce the impact of overfitting to a considerable degree and improve the reliability of phishing detection. Although the research had good performance, it mainly covered ensemble classifiers and did not investigate comparative studies in a broader range of models, nor did it consider the new paradigms of federated and large language model-based learning, which the current study deals with in order to expand scalability and flexibility to real-time phishing detection. Table 1 summarizes all the previous studies.

Table 1. Previous studies summary.

Ref	Dataset	Classifiers Used	Best Classifiers Accuracy
[21]	Kaggle (D1), Kaggle (D2), CatchPhish(D3), CatchPhish(D4), CatchPhish (D5), Ebbu2017 (D6)	NB, LR, SVM, kNN, J48, AdaBoost, MLP, RF	Random Forest with Kaggle datasets (96.25%)
[28]	Ebbu2017	RF, NB, kNN, Adaboost, K-Star, SMO, DT	Random Forest (97.98%)
[29]	Phishtankwebsite	NB, RF, SVM, XGBoost, Stacking	SVM (86.3%)
[30]	Malicious and Benign Webpages dataset	RF, NB, CNN, LSTM	Naive Bayes (96.01%)
[31]	Kaggle URL Phishing	DT, RF, NB, GBM, KNN, SVC, LSD (hybrid LR+SVC+DT)	LSD Hybrid (98.12%)
[32]	ham corpora	SVM, NN, SOM, K-means	SVM (97.99%)
[33]	Public dataset	Naive Bayes, Decision Trees (DT), Random Forest (RF), SVM, KNN, Multi-Layer Perceptrons (MLP), Gradient Boosting, and Linear Regression (LR).	LR (99.79%)
Our Study	ISCX-URL-2016	XGBoost, Random Forest, Bagging, AdaBoost, Gradient Boosting, CatBoost, LightGBM, Logistic Regression, ExtraTrees, SVM, Decision Tree, SGD, KNN, Passive Aggressive, Ridge Classifier, QDA, LDA, and HistGradientBoost	HistGradientBoost (98.47%)

3 URL Phishing Detection Model

The proposed model performs admirably as a comprehensive ML classifier for phishing attack detection. The model is used to assess the performance of several ML classifiers for phishing attack detection and determine the best one. Figure 1 shows the proposed model flowchart. The proposed model consisted of several stages; the next subsection discusses each stage:

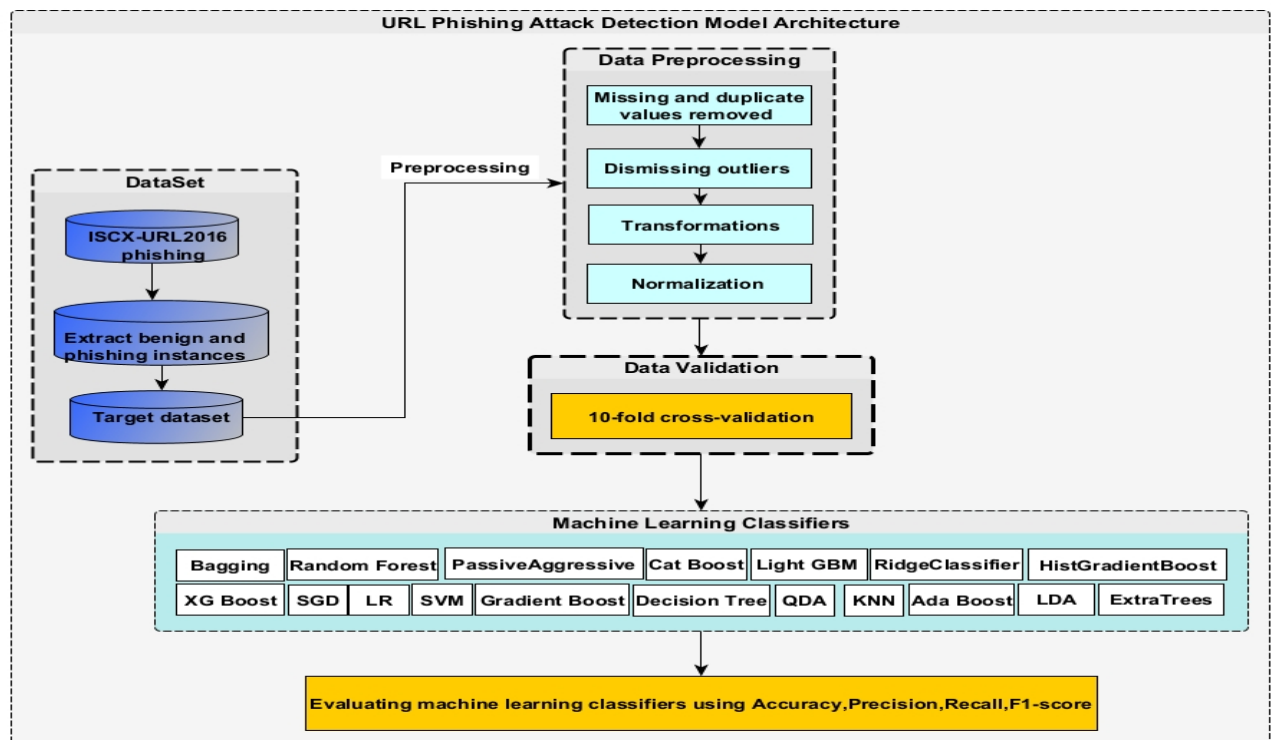


Figure 1. URL phishing detection model.

3.1 Dataset: ISCX-URL-2016 dataset

The ISCX-URL-2016 dataset [34] It is a comprehensive benchmark dataset created by the Canadian Institute for Cybersecurity for research in malicious URL detection. It contains a large collection of labeled URLs categorized as benign, phishing, malware, or spam. Designed to support the development and evaluation of ML models for cybersecurity applications, the dataset enables researchers to analyze patterns in URL structures and improve threat detection systems. Its diverse and well-labeled data make it a valuable resource for advancing cybersecurity research. So, this paper focuses on URL phishing attack detection; therefore, the benign phishing was extracted from the original dataset.

After carefully dataset cleaned by deleting the missing, redundant, and infinite values and 45,000 samples were left, including 35,000 benign URLs and 10,000 phishing URLs. Table 2 shows the 79 numerical and categorical features of lexical, host-based, and content-based features that were used to represent each instance of the URL. Label encoding was used to convert non-numeric attributes to numerical ones, and the min-max scaling method was used to normalize all the features so that all the values have equal distribution and there is no variation in model convergence. Since there was an imbalance in classes in the first place, 10-fold cross-validation was used as a powerful evaluation method. It is a technique that guarantees equal contribution of each subset of data to both model training and testing enhancement of generalization and reliability over fixed holdout splits. The distribution of ISCX-URL2016 labels is depicted in Figure 2.

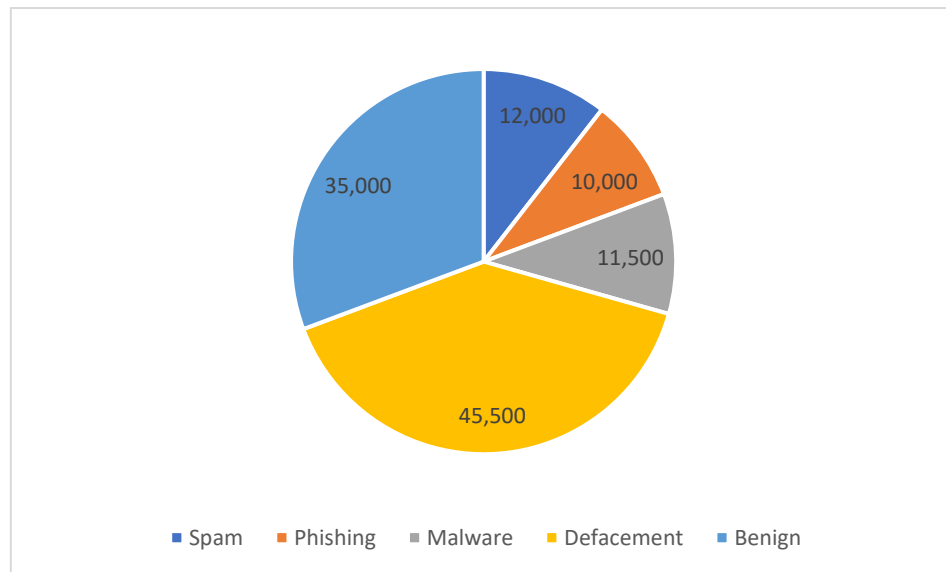


Figure 2. ISCX-URL2016 label distribution.

Table 2. Features of the dataset.

No	Feature	Category	No	Feature	Category
F1	Querylength	Lexical	F41	Directory_DigitCount	Lexical
F2	domain_token_count	Structural	F42	File_name_DigitCount	Lexical
F3	path_token_count	Structural	F43	Extension_DigitCount	Lexical
F3	avgdomaintokenlen	Structural	F44	Query_DigitCount	Lexical
F5	longdomaintokenlen	Structural	F45	URL_Letter_Count	Lexical
F6	avgpathtokenlen	Structural	F46	host_letter_count	Lexical
F7	tld	Structural	F47	Directory_LetterCount	Lexical
F8	charcompvowels	Lexical	F48	Filename_LetterCount	Lexical
F9	charcompacce	Lexical	F49	Extension_LetterCount	Lexical
F10	ldl_url	Lexical	F50	Query_LetterCount	Lexical
F11	ldl_domain	Lexical	F51	LongestPathTokenLength	Structural
F12	ldl_path	Lexical	F52	Domain_LongestWordLength	Structural
F13	ldl_filename	Lexical	F53	Path_LongestWordLength	Structural
F14	ldl_getArg	Lexical	F54	sub-Directory_LongestWordLength	Structural
F15	dld_url	Lexical	F55	Arguments_LongestWordLength	Structural
F16	dld_domain	Lexical	F56	URL_sensitiveWord	Lexical
F17	dld_path	Lexical	F57	URLQueries_variable	Lexical
F18	dld_filename	Lexical	F58	spcharUrl	Lexical
F19	dld_getArg	Lexical	F59	delimiter_Domain	Structural
F20	urlLen	Lexical	F60	delimiter_path	Structural
F21	domainlength	Structural	F61	delimiter_Count	Structural
F22	pathLength	Structural	F62	NumberRate_URL	Lexical
F23	subDirLen	Structural	F63	NumberRate_Domain	Lexical
F24	fileNameLen	Structural	F64	NumberRate_DirectoryName	Lexical
F25	this.fileExtLen	Structural	F65	NumberRate_FileName	Lexical

F26	ArgLen	Structural	F66	NumberRate_Extension	Lexical
F27	pathurlRatio	Structural	F67	NumberRate_AfterPath	Lexical
F28	ArgUrlRatio	Structural	F68	SymbolCount_URL	Lexical
F29	argDomanRatio	Structural	F69	SymbolCount_Domain	Lexical
F30	domainUrlRatio	Structural	F70	SymbolCount_Directoryname	Lexical
F31	pathDomainRatio	Structural	F71	SymbolCount_FileName	Lexical
F32	argPathRatio	Structural	F72	SymbolCount_Extension	Lexical
F33	executable	Structural	F73	SymbolCount_Afterpath	Lexical
F34	isPortEighty	Structural	F74	Entropy_URL	Lexical
F35	NumberOfDotsinURL	Structural	F75	Entropy_Domain	Lexical
F36	ISIpAddressInDomainName	Structural	F76	Entropy_DirectoryName	Lexical
F37	CharacterContinuityRate	Lexical	F77	Entropy_Filename	Lexical
F38	LongestVariableValue	Structural	F78	Entropy_Extension	Lexical
F39	URL_DigitCount	Lexical	F79	Entropy_Afterpath	Lexical
F40	host_DigitCount	Lexical	Lable	benign, malware, defacement, spam, and phishing	

3.2 Extract Benign and Phishing Instances

The ISCX-URL-2016 dataset consists of four classes of URLs, i.e., spam, malware, phishing, defacement, and benign. This work focuses on phishing detection, and therefore, the data pertaining to attack phishing URLs and benign URLs are extracted to form a new dataset called the target dataset.

3.3 Preprocessing

Preprocessing is a critical step in ML. This process helps improve model accuracy, efficiency, and generalization [35]. preprocessing techniques include data cleaning, Data Transformation, and Data Splitting [36] [37]. Before modelling, the raw dataset went through an extensive pre-processing pipeline to ensure quality, uniformity, and ML algorithm suitability . This preprocessing step is important in phishing detection tasks, since noise, missing, or inconsistent data may greatly reduce the generalization capacity of classifiers. The raw dataset is expressed by formula 1 :

$$D = \{(x_i, y_i) | i = 1, 2, \dots, N\} \quad (1)$$

Where $x_i \in \mathbb{R}^d$ represents the feature describing the URL and $y_i \in (0, 1)$ represents the class label (phishing or benign).

3.3.1 Handling Missing and Infinite Values

All missing entries, represented by unknown markers, as well as invalid numerical values, were replaced with *Not a Number* (NaN). Subsequently, any instance containing NaN values was removed from the dataset; the clean dataset is expressed by formula 2.

$$D' = \{(x_i, y_i) \in D \mid \forall j, x_{ij} \neq NaN\} \quad (2)$$

This preprocessing stage gives rise to a dataset that is complete and numerically consistent, thereby assuring that every feature vector that is used for training is valid and robust for any subsequent ML solutions.

3.3.2 Feature Encoding

The dataset was separated into a feature matrix $X \in \mathbb{R}^{n' \times d}$ and label vector $y \in \mathbb{R}^{n'}$, where $N' \leq N$ is the number of valid samples as expressed by formula 3 :

$$X = [x_1, x_2, \dots, x_n]^T, y = [y_1, y_2, \dots, y_{n'}] \quad (3)$$

Since some features were categorical, a label encoding function $f: C \rightarrow \mathbb{Z}$ was applied to map categorical values into discrete integers as expressed by formula 4:

$$x'_{ij} = f(x_{ij}), \forall x_{ij} \in C \quad (4)$$

where C denotes the set of categorical values. This transformation preserves information content while ensuring numerical compatibility with classifiers.

3.3.3 Class Label Transformation

Similarly, the target labels y were encoded into binary form as in formula 5:

$$y'_i = \begin{cases} 0, & \text{if } y_i = \text{benign} \\ 1, & \text{if } y_i = \text{phishing} \end{cases} \quad (5)$$

This representation enables the use of supervised learning algorithms in a binary classification setting.

3.3.4 Normalization and Scaling

To mitigate the effect of heterogeneous feature scales and ensure balanced contribution of all attributes during training, Min-Max normalization was applied [38]. Each feature x_j was linearly rescaled to the interval $[0,1]$ according to formula 6:

$$x^*_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)}, \forall x_j \in (1, 2, \dots, d) \quad (6)$$

Here, $\min(x_j)$ and $\max(x_j)$ represent the minimum and maximum values of the feature x_j , respectively. This transformation preserves the original distribution of features while bounding them to a uniform range, thereby improving both optimization stability and convergence speed of learning algorithms.

Through this preprocessing pipeline, the dataset was converted into a complete, consistent, and numerically well-defined one. These steps were necessary to remove noise, standardize representation, and enable fair comparison between different classifiers and learning algorithms.

3.4 10-Fold Cross Validation

To come up with a reliable and accurate approximation of model performance, this study used a 10-fold cross-validation (CV) method as opposed to a single train-test split like 80/20 or 70/30. The widely known supervised learning evaluation technique is cross-validation, since it reduces overfitting possibilities as well as offers a more generalized evaluation of predictive

capacity as compared to simple hold-out validation [39]. In a 10-fold CV, the data is randomly split into ten roughly similar and discontinuous subsets, called folds. At every iteration, nine folds are utilized to train the model, and the remaining fold is used to test the model. This is done ten times, such that each observation will be applied once to the validation process and nine times to the training process. The scores of individual performances of all folds are then averaged to produce a stable and holistic estimate of the effectiveness of the model. The mathematically stated meaning of cross-validated accuracy would take the form of formula 7.

$$\text{Accuracy}_{cv} = \frac{1}{10} \sum_{k=1}^{10} \text{Accuracy}(F_k) \quad (7)$$

where F_k denotes the k-th fold.

The 10-fold CV method also has several methodological benefits. First, it uses all the available information to the maximum, which is crucial when there is a potential imbalance of the classes of phishing and benign URLs. Second, it minimizes random data partition variability, which improves the statistical credibility and reliability of the reported results. Third, it is fairer and more consistent among all the seventeen classifiers measured and can be directly and meaningfully compared. In general, the use of the 10-fold cross-validation empowers the generalization of the results and makes them more valid. The combined measures in the form of Accuracy, Precision, Recall, and F1-score of all folds give the overall knowledge of the predictive capability and practicality of each model.

3.5 Machine Learning Classifiers

In this work, the consideration of wide range of ML algorithms was considered, each of which represents a specific family of learning strategies and has different strengths in identifying phishing URL attacks. The algorithms chosen are XGBoost, Random Forest [40], Bagging, AdaBoost, Gradient Boosting, CatBoost, LightGBM, Logistic Regression, Extra Trees, SVM, Decision Tree, SGD, KNN, Passive Aggressive Classifier, Ridge Classifier, QDA, Linear Discriminant Analysis (LDA), and HistGradientBoost.

3.5.1 Extreme Gradient Boosting (XGBoost)

XGBoost [41, 42] It is an optimized gradient boosting framework that constructs additive decision trees in a sequential manner while minimizing a differentiable loss function. Its objective function is given in formula 8:

$$L(\phi) = \sum_{i=1}^n l(y_i, y'_i) + \sum_{k=1}^K \Omega(f_k) \quad (8)$$

where l is the loss function, y'_i the prediction, and $\Omega(f_k)$ a regularization term to control model complexity. This regularization enhances generalization and prevents overfitting, making XGBoost particularly effective for phishing detection.

3.5.2 Random Forest (RF)

Random Forest [43, 44] It is an ensemble method that builds multiple decision trees on random subsets of data and features and aggregates their predictions through majority voting. The prediction for classification is given in formula 9:

$$y' = \text{mode}((x), h_1(x), \dots, h_m(x)) \quad (9)$$

where $h_j(x)$ is the prediction of the j-th decision tree. By reducing variance through averaging, Random Forest achieves high stability and robustness.

3.5.3 Bagging Classifier

Bootstrap Aggregating (Bagging) [45] generates multiple subsets of the training data using bootstrapping and trains weak classifiers (commonly decision trees) on them. The ensemble prediction is given in formula 10:

$$y^{\wedge} = \frac{1}{M} \sum_{m=1}^M h_m(x) \quad (10)$$

where M is the number of base learners. Bagging reduces overfitting by lowering prediction variance.

3.5.4 AdaBoost

Adaptive Boosting (AdaBoost) [46, 47] sequentially trains weak learners, adjusting the weights of misclassified instances at each iteration. The final strong classifier is a weighted combination that is given in formula 11:

$$H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x)) \quad (11)$$

where $h_t(x)$ is the weak learner and α_t its weight is based on classification error. AdaBoost emphasizes difficult samples, improving performance.

3.5.5 Gradient Boosting

Gradient Boosting [47, 48] builds additive models sequentially by optimizing a loss function via gradient descent. At each stage, a weak learner is fitted to the negative gradient of the loss function, as given in formula 12:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \quad (12)$$

where $F_m(x)$ is the ensemble at iteration m, $h_m(x)$ is the new base learner, and γ_m is the learning rate.

3.5.6 CatBoost

CatBoost[49] It is a gradient boosting algorithm designed to efficiently handle categorical features. It minimizes the following loss function with symmetric trees, which is given in formula 13:

$$L = \sum_{i=1}^n l(y_i, F(x_i)) + \sum_{t=1}^T \Omega(h_t) \quad (13)$$

where $F(x)$ represents the additive model. Its innovations in ordered boosting reduce target leakage and improve generalization.

3.5.7 LightGBM

LightGBM (Light Gradient Boosting Machine) [50] employs a leaf-wise tree growth strategy with gradient-based one-sided sampling (GOSS) to handle large datasets efficiently. The objective function is like gradient boosting, that is, given in formula 14:

$$L = \sum_{i=1}^n l(y_i, \hat{y}_i) + \lambda \|w\|^2 \quad (14)$$

with regularization λ to avoid overfitting.

3.5.8 Logistic Regression

Logistic Regression [51, 52] is a generalized linear model that estimates the probability of a binary class using the logistic function that is given in formula 15:

$$P(y = 1|x) = \frac{1}{1 + e^{-(w^T x + b)}} \quad (15)$$

where w and b are the parameters. Classification is based on a probability threshold

3.5.9 Extra Trees (Extremely Randomized Trees)

Extra Trees [53-55] builds an ensemble of decision trees where splits are chosen randomly rather than optimally. The prediction follows majority voting which is given in formula 16:

$$y' = \text{mode}(h_1(x), h_2(x), \dots, h_m(x)) \quad (16)$$

This increases diversity among trees, lowering variance and improving generalization.

3.5.10 Support Vector Machine (SVM)

SVM [56, 57] aims to find the optimal hyperplane that maximizes the margin between classes. The decision function is given in formula 17:

$$f(x) = \text{sign}(w^t x + b) \quad (17)$$

subject to constraints $y_i = (w^t x + b) \geq 1$. Margin maximization enhances classification robustness.

3.5.11 Decision Tree

A Decision Tree [58] recursively splits the data space using attribute-based conditions. The splitting criterion is typically based on Information Gain, as is given in formula 18:

$$IG(D, A) = H(D) - \sum_{v \in \text{Values}(A)} \frac{D_v}{D} H(D_v) \quad (18)$$

where $H(D)$ is the entropy of dataset D .

3.5.12 Stochastic Gradient Descent (SGD)

SGD [59] It is an optimization method for linear classifiers, such as linear SVM or logistic regression. The update rule is given in formula 19:

$$w^{(t+1)} = w^{(t)} - \eta \nabla l(w^{(t)}; x_i, y_i) \quad (19)$$

where η is the learning rate.

3.5.13 K-Nearest Neighbors (KNN)

KNN [60, 61] is a non-parametric algorithm that assigns a label based on the majority class among the k nearest neighbors in the feature space, as given in formula 20:

$$\hat{y} = \text{mode}(y_i: x_i \in N_k(x)) \quad (20)$$

where $N_k(x)$ is the set of k nearest neighbors.

3.5.14 Passive-Aggressive Classifier

The Passive-Aggressive algorithm [60] updates weights only when a misclassification occurs. The update rule is as given in formula 21:

$$w_{t+1} = w_t + T_t y_t x_t \quad (21)$$

where

$T_t = \frac{\max(0, 1 - y_t w_t^T x_t)}{\|x_t\|^2}$. This ensures aggressiveness for errors while remaining passive for correct predictions.

3.5.15 Ridge Classifier

Ridge Classifier applies L2 regularization to logistic regression or linear classification. The optimization objective is given in formula 22:

$$\min_w \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \|w\|_2^2 \quad (22)$$

where λ is the regularization parameter.

3.5.16 Quadratic Discriminant Analysis (QDA)

QDA [62] assumes each class follows a Gaussian distribution with its own covariance matrix. The discriminant function is given in formula 23:

$$\delta_k(x) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (x - u_k)^T \Sigma_k^{-1} (x - u_k) + \ln \pi_k \quad (23)$$

Where u_k , Σ_k , and π_k are the mean, covariance, and prior of class k .

3.5.17 Linear Discriminant Analysis (LDA)

LDA [63] assumes Gaussian class distributions with shared covariance. The discriminant function is given in formula 24:

$$\delta_k(x) = x^T \Sigma^{-1} u_k - \frac{1}{2} u_k^T \Sigma^{-1} u_k + \ln \pi_k \quad (24)$$

where Σ is the common covariance matrix.

3.5.18 Histogram-Based Gradient Boosting (HistGradientBoost)

HistGradientBoost discretizes continuous features into histograms to accelerate gradient boosting. The model is built additively as in formula 25:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \quad (25)$$

where histograms approximate the gradient statistics.

3.6 Proposed Model Properties & Positioning

The proposed phishing URL detection model is presented with some properties that place it in the right position in the greater context of the ML-based cybersecurity solutions. First, the system has data-centric characteristics based on ISCX-URL-2016 benchmark data, and benign URL and phishing URL are mined systematically that the real-world attack patterns are covered. In the preprocessing step, normalization and feature scaling are performed to ensure stability and comparability of the models, thus allowing a fair comparison between different algorithms. Second, the model uses 10-fold cross-validation that enhances the performance estimates' reliability as it reduces bias and variance. This makes sure that the results are not based on a specific division of the dataset but are based on generalized detection capability. Third, the paper puts itself by using the wide variety of ML algorithms including instance-based algorithms (KNN), linear algorithms (logistic regression, ridge classifier, SGD), probabilistic algorithms (LDA, QDA), and ensemble learner (XGBoost, LightGBM, CatBoost, HistGradientBoost, Random Forest). This makes it possible to perform a strong comparative analysis that enables one to note the advantages and disadvantages of the different families of classifiers in the domain of detecting URL phishing. Lastly, the positioning of the model is feasible: the models are not only considered in terms of accuracy, but also precision, recall, and F1-score are important to balance the false positives and false negatives. This puts the work as a step to operationally reliable URL phishing detection systems, which can be implemented in real-life security structures.

3.7 Implementation Environment

All the experiments were performed in Python 3.13 on a workstation with an Intel Core i7 processor (3.2 GHz), 16 GB RAM, and Windows 11 (64-bit). The classical models used in the ML were carried out with the help of Scikit-learn (v1.6.1) library, and the XGBoost, LightGBM, and CatBoost were applied to the advanced methods of the ML ensemble and gradient boosting methods. Other data processing and visualization libraries that assisted in this study were NumPy, Pandas, Matplotlib, and Seaborn. To achieve the reproducibility of the experiments a fixed random seed was used, and the models were tested with a 10-Fold Cross-Validation strategy to get statistically stable and objective results of the performance. All models were trained under identical preprocessing and parameter settings to ensure fairness in comparison. Table 3: Summary of ML Algorithms and their Python library.

Table 3. Used ML algorithms and their Python library.

Category	Algorithm(s)	Library Used
Ensemble Learning Methods	XGBoost, LightGBM, CatBoost, AdaBoost, GradientBoosting, HistGradientBoost, Bagging, RandomForest, ExtraTrees	XGBoost, LightGBM, CatBoost, Scikit-learn
Linear Models	Logistic Regression, Ridge Classifier, Stochastic Gradient Descent (SGD), Passive-Aggressive Classifier	Scikit-learn
Discriminant Analysis Models	Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA)	Scikit-learn

Instance-Based Model	K-Nearest Neighbors (KNN)	Scikit-learn
Tree-Based Model	Decision Tree	Scikit-learn
Support Vector Machine	Support Vector Classifier (SVM)	Scikit-learn

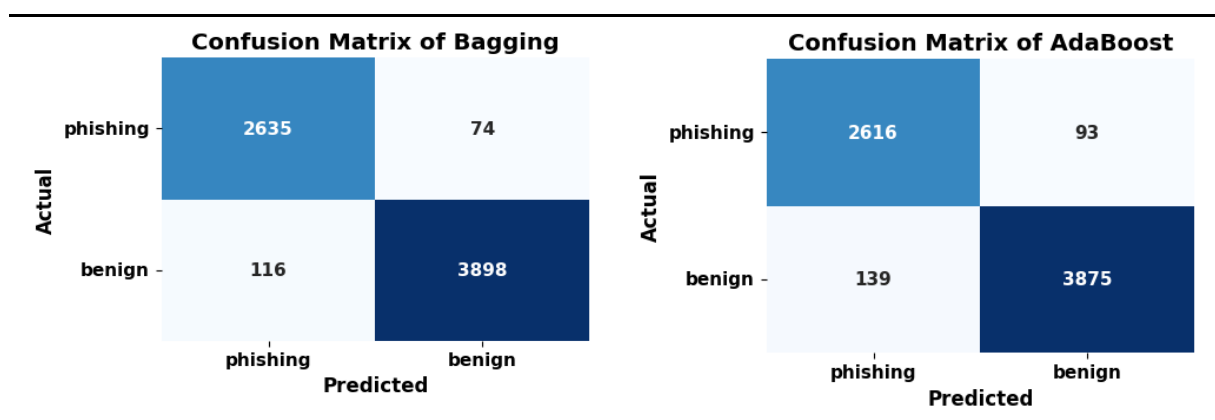
Table 4 shows the Hyperparameters of each classifier.

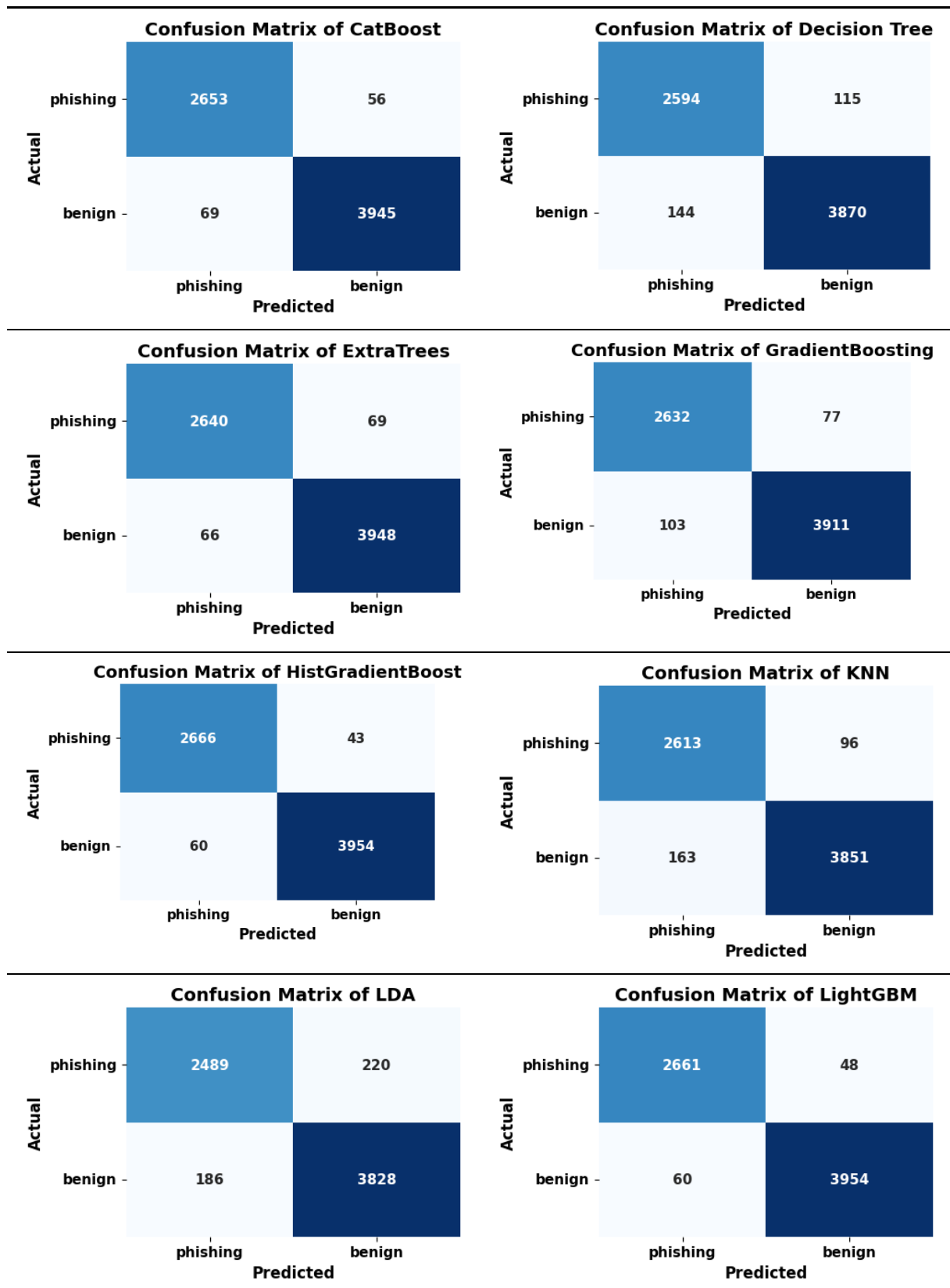
Table 4. Hyperparameter settings of the classifier.

Model	Hyperparameters Setting
HistGradientBoost	histogram-based trees; learning_rate=0.1; max_depth=auto; n_estimators=100
LightGBM	boosting=gbdt; num_leaves=31; max_depth=-1; n_estimators=100
XGBoost	booster=gbtrees; max_depth=6; learning_rate=0.1; subsample=1.0; colsample_bytree=1.0; n_estimators=100
CatBoost	depth=6; learning_rate=auto; l2_leaf_reg=3; iterations=100; loss_function=Logloss; verbose=0
RandomForest	n_estimators=100; max_depth=None; max_features='sqrt'
ExtraTrees	n_estimators=100; max_depth=None; max_features='sqrt'
GradientBoosting	n_estimators=100; max_depth=3; learning_rate=0.1
AdaBoost	n_estimators=50; base_estimator=DecisionTree(max_depth=1)
Bagging	n_estimators=10; base_estimator=DecisionTree()
SVM (RBF)	C=1.0; gamma='scale'
Logistic Regression	penalty='l2'; C=1.0; solver=lbfgs; max_iter=1000
Ridge Classifier	alpha=1.0
SGD	loss='log_loss'; penalty='l2'; max_iter=1000
Passive-Aggressive	C=1.0; loss='hinge'
Decision Tree	criterion='gini'; max_depth=None
KNN	n_neighbors=5; metric='minkowski'; weights='uniform'
LDA	solver='svd'
QDA	reg_param=0.0

4 Results and Discussion

The phishing attacks have been tested with different ML classifiers, XG Boost, Ada Boost, Gradient Boost, Cat Boost, SVM, Random Forest, Decision Tree, SGD, Light GBM, KNN, and ANN using the ISCX-URL-2016 dataset. The confusion matrix obtained from the experiment for each classifier is shown in Figure 3.





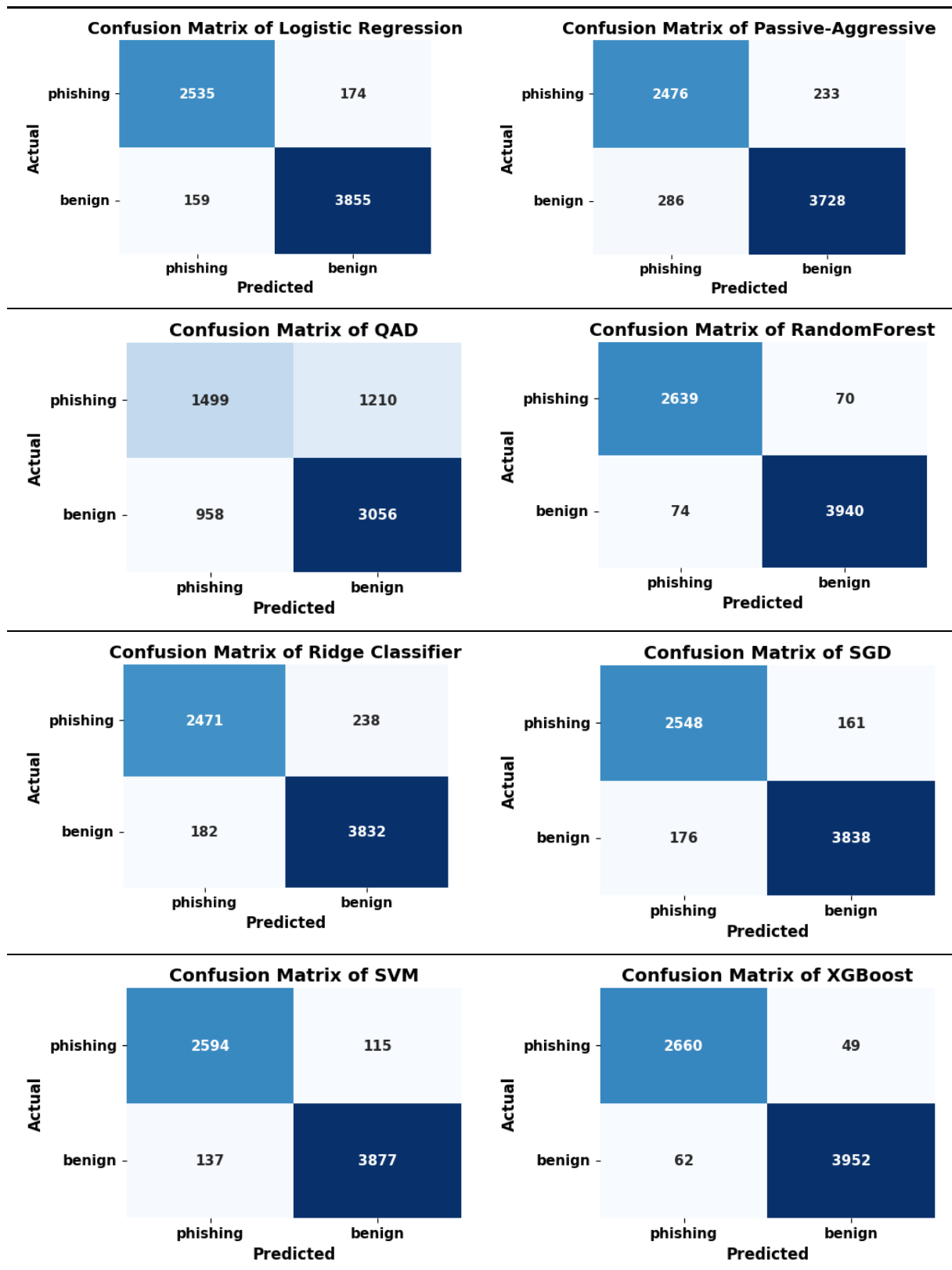


Figure 3. Confusion matrix of each classifier.

Table 5 shows the accuracy, precision, sensitivity, F-score, Training, Prediction time, and complexity of each classifier.

Table 5. Obtained results.

Model	Accuracy	Precision	Recall	F1-score	Train Time	Pred Time (ms)	Complexity
XGBoost	98.4 ± 0.8	98.3 ± 0.8	98.3 ± 0.8	98.3 ± 0.8	0.173 ± 0.007	0.003 ± 0.000	100 trees (max depth = 6)
RandomForest	97.9 ± 0.5	97.8 ± 0.6	97.8 ± 0.5	97.8 ± 0.6	0.999 ± 0.060	0.018 ± 0.003	100 trees (adaptive depth)
Bagging	97.2 ± 0.7	97.0 ± 0.8	97.2 ± 0.7	97.1 ± 0.7	1.168 ± 0.080	0.007 ± 0.002	10 estimators (DT base)
AdaBoost	96.6 ± 0.9	96.3 ± 1.0	96.6 ± 0.9	96.4 ± 1.0	0.883 ± 0.168	0.019 ± 0.004	50 stumps (depth = 1)
GradientBoosting	97.3 ± 0.6	97.2 ± 0.7	97.3 ± 0.6	97.2 ± 0.7	3.610 ± 0.171	0.003 ± 0.001	100 trees (max depth = 3)
CatBoost	98.1 ± 0.5	98.0 ± 0.5	98.1 ± 0.5	98.1 ± 0.5	10.530 ± 1.665	0.008 ± 0.001	100 trees (depth = 6)
LightGBM	98.4 ± 0.7	98.3 ± 0.7	98.4 ± 0.7	98.3 ± 0.7	0.208 ± 0.006	0.005 ± 0.001	100 trees (leaf-wise, depth adaptive)
LogisticRegression	95.1 ± 0.9	94.9 ± 0.9	94.8 ± 0.9	94.9 ± 0.9	0.050 ± 0.007	0.002 ± 0.000	Params ≈ 78
ExtraTrees	98.0 ± 0.6	97.9 ± 0.7	97.9 ± 0.6	97.9 ± 0.6	0.502 ± 0.020	0.022 ± 0.002	100 trees (adaptive depth)
SVM	96.3 ± 0.8	96.1 ± 0.9	96.2 ± 0.8	96.1 ± 0.8	0.467 ± 0.039	0.180 ± 0.030	SV-based (≈ hundreds)
DecisionTree	96.2 ± 0.6	95.9 ± 0.7	96.1 ± 0.6	96.0 ± 0.6	0.154 ± 0.012	0.001 ± 0.000	Depth ≈ 8–20
SGD	95.0 ± 0.9	94.8 ± 0.8	94.8 ± 1.1	94.8 ± 0.9	0.068 ± 0.011	0.002 ± 0.000	Params ≈ 78
KNN (k=5)	96.2 ± 0.7	95.9 ± 0.8	96.2 ± 0.7	96.0 ± 0.8	0.013 ± 0.001	0.029 ± 0.006	k = 5
PassiveAggressive	92.3 ± 5.6	91.9 ± 4.8	92.1 ± 4.9	92.0 ± 5.5	0.035 ± 0.013	0.002 ± 0.000	Params ≈ 78
RidgeClassifier	93.7 ± 1.0	93.7 ± 1.0	93.3 ± 1.1	93.5 ± 1.0	0.014 ± 0.002	0.002 ± 0.000	Params ≈ 78
QDA	67.8 ± 2.8	66.3 ± 4.2	65.7 ± 3.7	65.9 ± 3.0	0.037 ± 0.006	0.002 ± 0.000	Closed-form DA (per-class covariances)
LDA	94.0 ± 1.2	93.8 ± 1.2	93.6 ± 1.2	93.7 ± 1.2	0.033 ± 0.003	0.002 ± 0.000	Closed-form DA
HistGradientBoost	98.5 ± 0.7	98.4 ± 0.7	98.5 ± 0.7	98.4 ± 0.7	0.784 ± 0.088	0.010 ± 0.002	HistGB (adaptive depth)

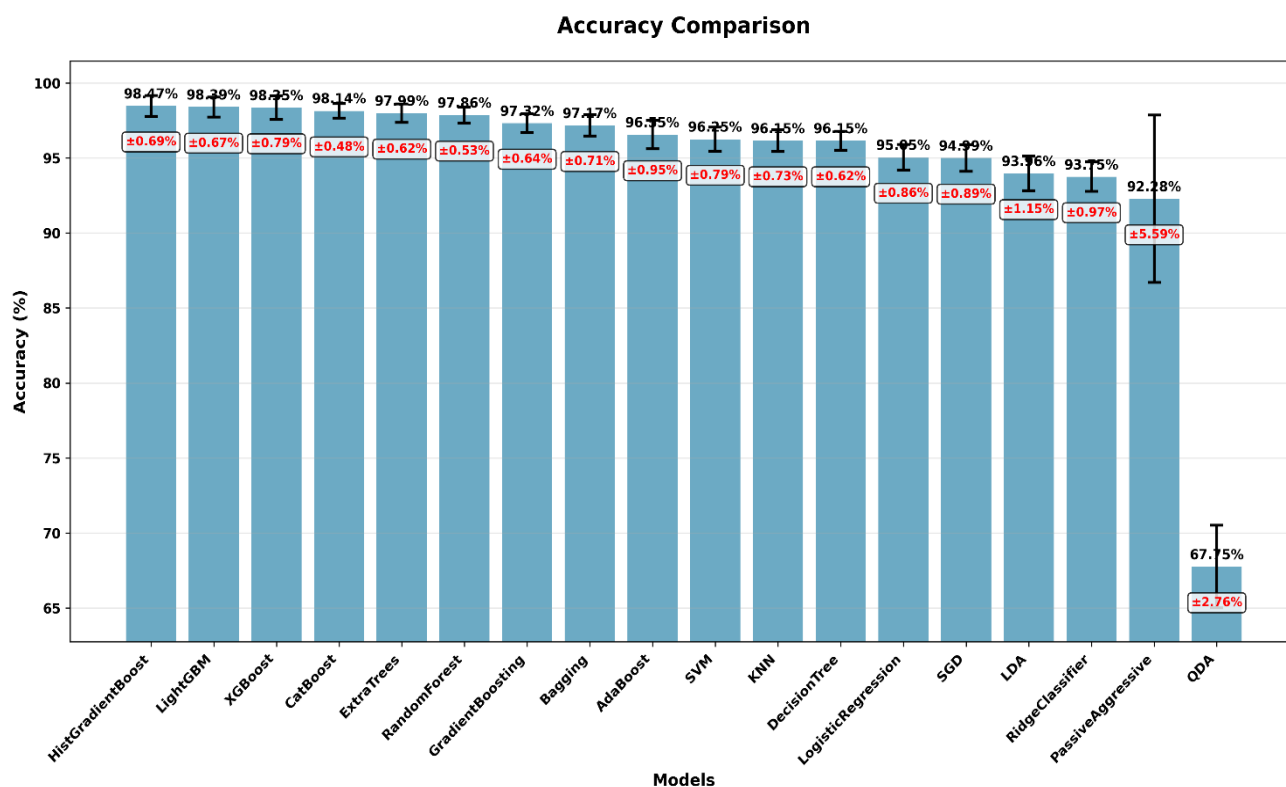


Figure 4. Accuracy results.

The accuracy of the various classifiers used for URL phishing attack detection is illustrated through the comparison of different classifiers in figure 4. Based on experimental results, it is demonstrated that ensemble-based models perform significantly better than today's classifiers in phishing URL detection. The result shows that the boosting algorithms are effective in demonstrating the complexity and nonlinearity of phishing URLs, with HistGradientBoost being the most accurate (98.47%), followed by LightGBM (98.39%) and XGBoost (98.35%). CatBoost (98.14%), ExtraTrees (97.99%) and RandomForest (97.86%) did well as well, highlighting the power of ensemble learning techniques. The moderate performers were the ML classifiers namely Logistic Regression (95.05%), SVM (96.25%) and KNN (96.15%), while the margin-based classifiers failed to perform as well as these, namely PassiveAggressive (92.28%) and RidgeClassifier (93.75%). The lowest accuracy result was obtained for QDA (67.75%) possibly due to the assumption of Gaussian distribution which is not very close to the irregular phishing URLs. In conclusion, it can be said that the most successful methods of boosting indicate the best detection; furthermore, traditional linear models are not always effective in extrapolating into the multidimensional space of phishing sites. The results herein imply that ensemble learning and especially the gradient boosting techniques are a promising path to develop powerful and effective phishing detection systems.

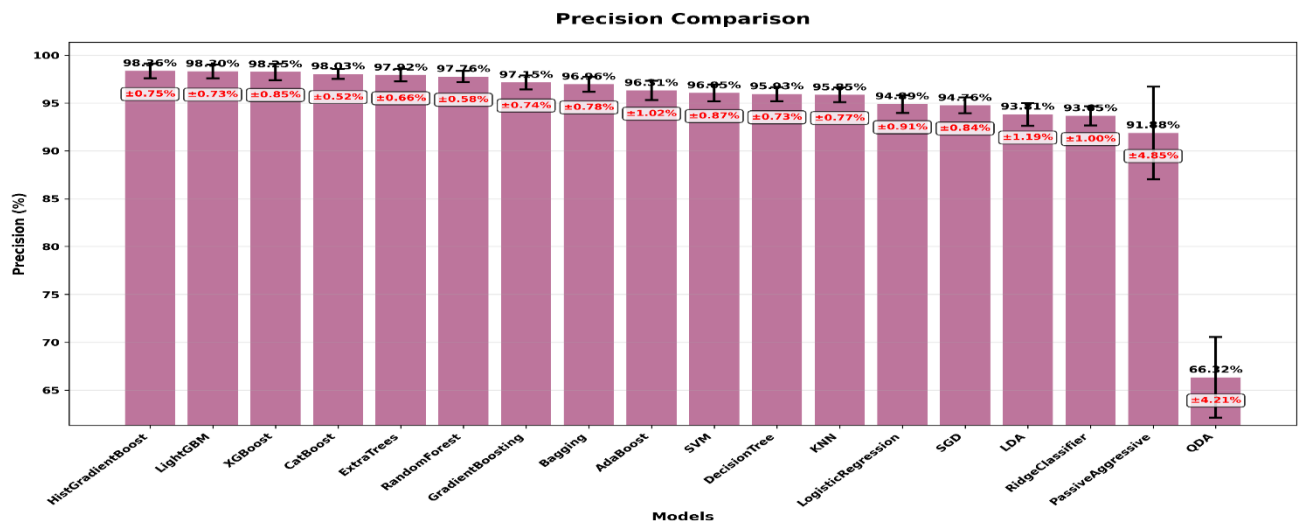


Figure 5. Precision results.

Different ML classifiers' precision is displayed on Figure 5. These precise results also validate the effectiveness of ensemble and boosting algorithms in phishing URLs. HistGradientBoost had the best precision values with 98.36%, it is close to LightGBM with 98.30%, and XGBoost, with 98.25%, meaning that they have the lowest rate of false positives and can reliably detect legitimate URLs against phishing attempts. Also, high precision values were obtained for CatBoost (98.03%), ExtraTrees (97.92%) and RandomForest (97.76%), which proved the effectiveness of tree-based ensembles in the case of capturing subtle features from URL. The moderate performance of the rapid classifiers (Logistic Regression (94.89%), SVM (96.05%) and KNN (95.85%)) indicates that the linear and distance-based classifiers were not as strong as the boosting methods in this case, which shows the compromising nature of these classifiers in dealing with complex patterns of features. The results of PassiveAggressive (91.88), RidgeClassifier (93.65) were less accurate than margin-based, while QDA has the lowest score (66.32); this encourages to dismiss the use of the algorithm in this area of problems. Generally, the outcomes suggest that boosting-based ensembles not only demonstrate the highest level of accuracy, but they also give high precision as well, making them the most effective choice for real-world phishing detection scenarios where minimizing false alarms is a critical concern.

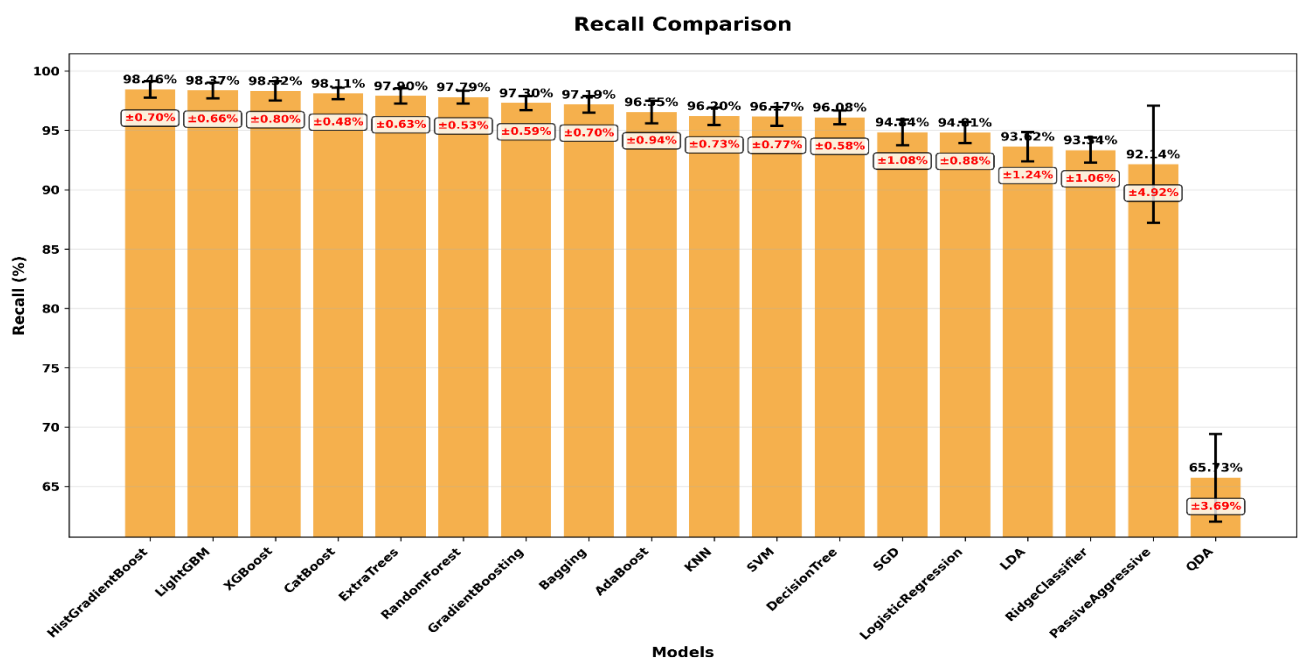


Figure 6. Recall results.

The recall (sensitivity) of different ML classifiers is shown in Figure 6. The recall of different models shows that the boosting-based ensemble algorithms outperform other algorithms in the detection of phishing URLs, and the better model is HistGradientBoost with the recall of 98.46%, followed by LightGBM with 98.37%, and XGBoost with 98.32%. It was concluded that these models can be more effective in correctly identifying phishing URLs with fewer false negative (FN) results, which is a crucial requirement in cybersecurity when a phishing threat goes undetected, it can prove fatal. Ensemble learning has also been proven to be a good detection coverage, with CatBoost (98.11%), ExtraTrees (97.90%) and RandomForest (97.79%) also performing well. On the other hand, Logistic Regression, SVM and KNN had a moderate recall (94.81 percent, 96.17 percent and 96.20 percent respectively) while PassiveAggressive and RidgeClassifier were poor in their recall (92.14 percent and 93.34 percent respectively). Again, the poorest came from QDA (65.73%), which wasn't well suited for the irregular and non-linear nature of the phishing URL patterns. In general, the results show that the models that boost them not only achieve high precision but also high recall in which most phishing attacks are successfully identified.

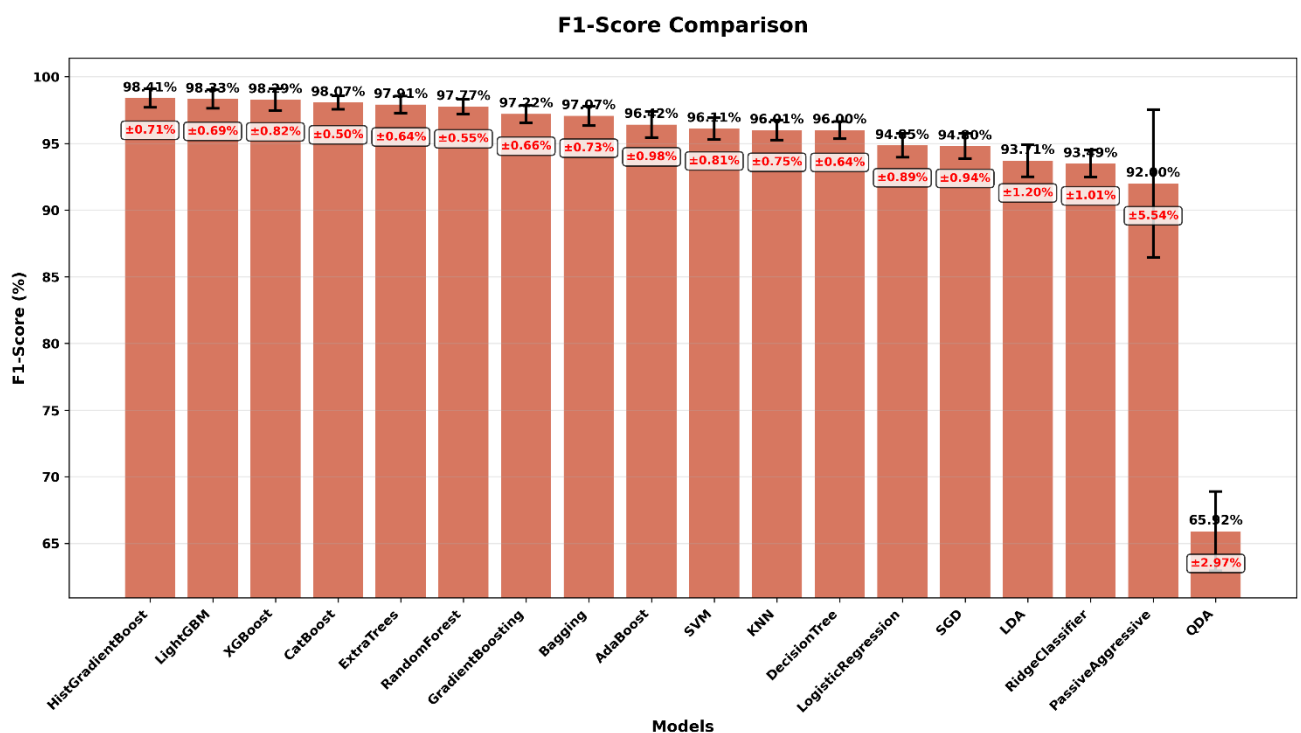


Figure 7. F1-score results.

Figure 7 shows the F-score of the different ML classifiers. The results of the F1-score give an overall picture of the precision and recall, and again, the excellence of the ensemble-based approaches is proven in detecting phishing URLs. The highest F1-score (98.41%), then LightGBM (98.33%), XGBoost (98.29%), and CatBoost (98.07%), allowed us to determine the strong ability to guarantee high precision along with high recall rates with boosting algorithms. ExtraTrees had the highest F1-score of 97.91% followed by RandomForest (97.77%), Bagging (97.07%), and GradientBoosting (97.72%), which illustrates the reliable nature of the ensemble approaches. Contrastingly, more conservative classifiers like Logistic Regression (94.85%), SVM (96.11%), DecisionTree (96.00%), and KNN (96.01%) performed well and yet never surpassed boosting models, whereas margin-based classifiers like PassiveAggressive (92.00%) and RidgeClassifier (93.49%) failed. The weakest result was once again that of QDA (65.92%), which made it inappropriate for phishing detection. Overall, the results indicate that the boosting-based ensembles provide an optimal trade-off between precision and recall; thus, they are the strongest and most reliable models of actual phishing detection systems.

Figure 8 shows a comparative evaluation of the model efficiency between training and prediction time and gives some details as to their applicability to real-time URL phishing detection. The results indicate a definite performance trade-off within the family of classifiers. The highest computational cost was CatBoost, which took the longest time to train since it uses iterative boosting and categorical encoding, and then came Gradient Boosting and Random Forest, which used relatively long training times also due to the large number of trees to be built in constructing the model. Linear models like SGD, Passive-Aggressive, and Logistic Regression, on the other hand, trained very quickly, taking only fractions of a second, and were popular with large-scale or frequently retrained systems. Decision Tree and KNN were the slowest in terms of prediction time, the former being slow because of deep tree traversal, and the latter, being an instance-based algorithm, being slow because each query needs to compute distances. Linear models once again performed the quickest inference, but closely behind it were LightGBM, HistGradientBoost, and XGBoost, which provide a good tradeoff between performance and accuracy. These findings indicate that the boosting-based models offer a powerful tradeoff between deployment, i.e., with high accuracy and low inference latency, and linear models are very scalable to the edge or high-throughput scenarios where computational resources are small.

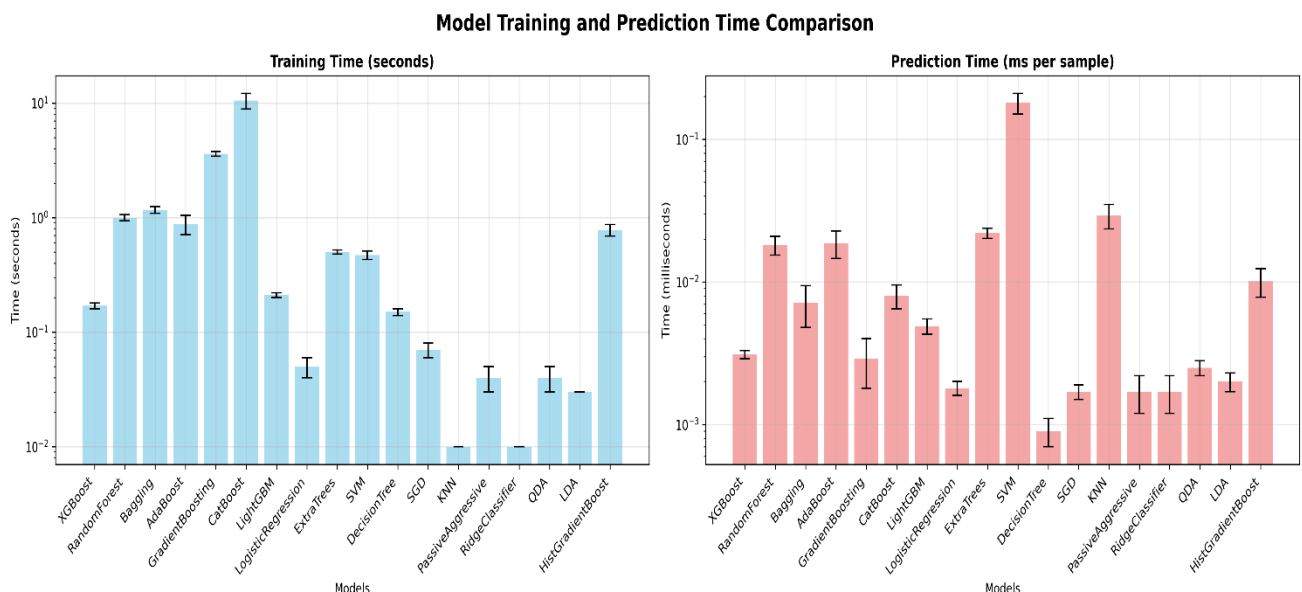


Figure 8. Training and prediction time.

Averaged across the experiments, ensemble-based models, particularly gradient boosting methods such as HistGradientBoost, LightGBM and XGBoost, outperform all the other models on all metrics. The radar plot in Figure 9 depicts the top 6 models with high accuracy, precision, recall and F1 scores of >98%. These models are also able to support the complexity and non-linearity of phishing URLs and have a high rate of false negative and high detection rate. The complexity obtained results in Table 4 will provide useful insights into the computational aspects of the classifiers considered and to their deployability in different environments. Ensemble model boosting has moderate to high model complexity as trees are constructed sequentially, but they are compact enough for speedy inference. Indicatively, HistGradientBoost, LightGBM and XGBoost have an effective tradeoff between depth and number of boosted trees to keep the models highly generalisable while keeping the memory footprint small. Tree based bagging methods such as ExtraTrees and Random Forest have a higher structural complexity, both employ many trees independently trained at varying depths, size of each tree is different, and hence larger model size and storage requirements. However,

the linear models (Logistic Regression, Ridge, SGD and Passive-Aggressive) have very few trainable features (around 79) and can be used in resource or edge security applications. KNN has low training costs, high inference complexity because of the use of computing the distance between the stored instances and itself. Overall, the comparison suggests that a boosting-based model offers the most complex to perform trade-off that is feasible to deploy a phishing URL detection system. Overall, these findings suggest that the boosting-based ensembles are the most effective and reliable approaches to solving phishing URL detection problems, offering a glimpse of how to build realistic and effective cyber security systems. It is essential to situate the results in the context of the new trends in cybersecurity, that is, Federated/Decentralized Learning and LLM-driven detection.

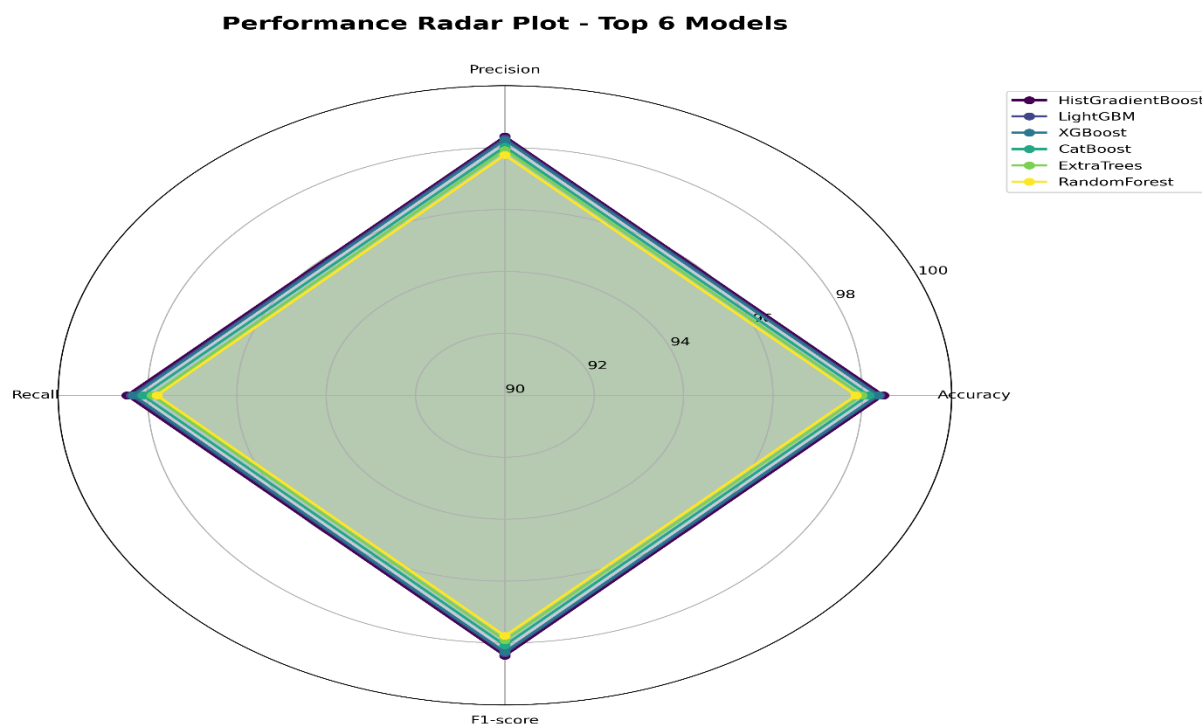


Figure 9. Radar plot for the top 6 models.

FL provides the opportunity to train a model to detect phishing without needing to gather all the data on one point (browser, enterprise gateway, Internet of Things, etc.) [64]. This paradigm explicitly addresses the problems relating to privacy, ethics of data and regulatory norms and enables the models to respond flexibly to the local attack patterns. The FL-based approaches would allow the ISCX-URL-2016 knowledgebase and other similar corpora to be growing continuously in a distributed fashion. It suffers from disadvantages such as communication overheads, non-independent and identically distributed (non-IID) client data, and being sensitive to adversarial updates (Poisoning attacks).

However, LLM-based detection [65] is based on the contextual reasoning ability of transformer-based architectures (such as GPT, BERT), which can assess the structure of the URL, the metadata information associated with it, and the semantic information contained within the website. While feature-based structured models can be tested in this paper, the advantage of semantic generalization of LLMs is that new phishing techniques that differ from the feature distribution used in training can be identified (not necessarily following the feature distribution in the training data). They suffer from computational costs, interpretability, and security dataset fine-tuning problems, however. Lastly, the present results validate that

ensemble methods such as LightGBM, HistGradientBoost and XGBoost result in optimal accuracy performance in centralized mode (98%-99%). Implementation of FL-based decentralized system would strengthen privacy-assuring delivery while LLM-based detection would enhance the level of protection against language and semantic inferencing-based zero-day phishing attacks.

5 Trade-offs, Threats, and Roadmap

The results prove the effectiveness of ensemble learning methods, particularly LightGBM, HistGradientBoost and XGBoost, which achieved high results of more than 98% accuracy. These results confirm that a classical ML-based classifier, trained on high quality data, like ISCX-URL-2016, can offer very strong URL phishing detection. When applied to a real-world cyber security environment, however, some trade-offs, potential threats and action steps must be considered individually.

5.1 Trade-offs

One of the most significant compromises is that between accuracy and efficiency. The ensemble model is very accurate but might require additional computing power in terms of training and inference compared to a light-weight classifier, such as logistic regression or SGD. These costs have been decreased by the modern servers and cloud infrastructures, but the additional computational overhead may not be feasible by the edge devices (e.g., personal browsers, IoT nodes). Also, there is a trade-off between the interpretability of the model and its complexity. In regulatory or forensic applications other models like decision trees or logistic regression are easier to understand, while gradient boosting is still accurate but relatively unclear. This brings up questions on the observance of new AI regulations governing models, with the focus on explainability [66].

5.2 Threats and Challenges

Despite the high baseline performance obtained with centralized training using a benchmark dataset, phishing attackers keep changing their tactics. Unless they are properly designed, static models are prone to degradation with time, especially in the case of zero-day attacks or in the case of obfuscation to defeat known distributions of features. The other threat is due to data imbalance and drift in the data. Contrary to benchmark datasets, real-world phishing data might be skewed and thus biased towards the majority classes when training classifiers. Lastly, data poisoning or adversarial examples are also a major challenge that causes adversarial manipulation. Attackers may provide misguided information during training or may design URLs that are formed in a specific manner so that they do not attract attention but still act maliciously.

5.3 Integration of ML, FL, and LLMs for URL Detection

To make deployment to the real world a reality, we deploy the most optimal ensemble model (e.g., HistGradientBoost/LightGBM/XGBoost) as an instance of a low-latency inline filter by browser extensions, email gateways, or secure web gateways (SWGs). Every requested URL is scored in milliseconds and vectorized, confident results are generated, allowing/block/warn actions to be taken, features and decisions are recorded in the security stack (API gateway, SIEM/SOAR) to be audited, and give feedback. To achieve privacy-aware continual learning, we use Federated Learning (FL) to train at training time: sites or endpoint cohorts fine-tune local models on their recent web traffic and transmit encrypted parameter deltas (not URLs of specific web pages) to a secure aggregator to be weighted averaged and redistributed globally.

This maintains the data regulations and follows the regulatory limitations and the changing campaigns. We add an LLM escalation path to the critical latency path to make cases uncertain or novel more resilient: samples that are marked as uncertain/high-risk are provided with contextual signals (e.g., adjacent email text or page snippets) and sent to an LLM service to conduct semantic reasoning and an explanatory verdict, rate limiting, and controlled calls to limit costs and throughput. It has a model registry and CI/CD (versioning, rollback, drift/poisoning monitors), and explainable AI artifacts (feature attributions, reasons why escalation) are made visible to analysts. The hybrid design is a tradeoff between accuracy (ML), privacy (FL), and contextual robustness (LLM) when it comes to the inline performance.

5.4 Future Directions and Roadmap

To tackle these challenges, a multi-pronged roadmap must be developed. The first step should be introducing privacy-preserving and decentralized learning, such as FL, in which distributed clients collaboratively learn detection models while not sharing any sensitive local data. It would facilitate the phishing detectors to stay updated with the local and global threat panorama and conform with privacy rules and regulations. Second, adaptive learning structures that can adapt or retrain online in reaction to changes in the dataset and attacker tactics are needed.

Enhancing the practical relevance of the proposed approach, it is worth mentioning how the models considered in this work can be applied to real security settings. When deployed normally, the trained phishing-URL classifier can be inserted on the decision path of the existing defenses, browser extensions, secure web gateways, email filtering systems, or API-based URL reputation services to such an extent that all requested URLs pass through the classifier before being rendered or delivered to the user. Since some of the ensemble models used in this paper (e.g., HistGradientBoost, LightGBM, XGBoost) were both very accurate and low-inference time, they can be deployed in such inline applications to make allow/block/warn decisions in real-time, whereas more lightweight linear models can be deployed to endpoint agents or mobile devices with smaller computational budget constraints.

An extension of centralized deployment is to enable privacy-aware and distributed environments with the help of federated learning. In this context, local training or fine-tuning of the phishing detector on an organization or edge device traffic can be done, and the latest model versions can be sent to an aggregator in the center, thus enhancing the global model without revealing raw URLs or user information. This resolves directly regulatory and confidentiality limitations and yet enables the detector to adjust to the newly appeared phishing attacks. Simultaneously, large language models (LLMs) can be added as an additional layer to situations when lexical or structural information is inadequate (e.g. to comprehend contextual information in emails, landing pages or obfuscated URLs). A hybrid pipeline whereby a lightweight ML classifier is used to perform the high-volume, low-latency filtering and an LLM is called on only ambiguous or high-risk samples, provides a useful accuracy-cost trade-off in practice deployments. Last, the application of explainable AI methods to this pipeline would assist security analysts in understanding the rationale behind a URL flag, which will instill trust and simplify integration with SOC processes. By doing so, the suggested ML-based detector can be gradually expanded to FL- and LLM-conscious phishing protection systems without altering the fundamental modeling approach.

6 Conclusion

The research results also suggest the importance of the development of the most advanced ensemble learning methods to cope with the problem of phishing URL recognition. Although

conventional classifiers like Logistic Regression, SVM, and KNN offered moderate performances, they were always lagging behind the performance of the ensemble-based counterparts since they were shown to be weak when it comes to dealing with highly irregular phishing patterns. Of the algorithms tested, boosting frameworks were always convenient models, with HistGradientBoost, LightGBM, and XGBoost providing a high level of precision, recall, and F1-scores, all over 98%. This consistency in various measures of evaluation highlights how the boosting techniques can support a tradeoff between detection coverage and low false alarm rates, which is highly critical in practical cybersecurity applications where missed detections and false positives are both potentially dangerous. Another observation of the analysis is that tree-based ensembles like the Random Forest and Extra Trees, although not as effective as boosting methods, performed quite well, implying that they could be a better choice where computational effectiveness and interpretability are concerned, rather than marginal performance improvement. Conversely, other algorithms, such as QDA, showed serious weaknesses, which strengthens the fact that the models that require high distributional assumptions cannot be applied in this field. Generally, this research highlights that boosting-based ensembles provide a strong base in phishing detection with both high predictive accuracy and scalability to complex URL architecture, thus creating a bright prospect in the development of reliable and scalable cybersecurity tools.

DECLARATIONS

Funding. "This research received no external funding"

Conflicts of interest. "The authors declare no conflicts of interest"

Ethics approval. "Not applicable. This study did not involve human participants, animals, or personally identifiable data; it relied solely on the publicly available ISCX-URL2016 benchmark dataset."

Informed consent. "Not applicable, as no human subjects were involved in this study."

Data and code availability. "The ISCX-URL2016 dataset analyzed in this study is publicly available from the Canadian Institute for Cybersecurity. The code supporting the findings is available from the corresponding author upon reasonable request."

Author contributions. "O. Almomani and M. A. Almaiah: Conceptualization, Supervision, M. Madi: Writing – review & editing. O. Almomani :Methodology, A. Shalalkeh: Software. M. Madi and A. Shalalkeh Writing – original draft. A. S. A. Shatat: Formal analysis, Validation. M. A. Almaiah: Investigation, Data curation. M. Madi: Writing – review & editing. M. K. Alsmadi: Validation, Visualization. I. M. Dweib: Investigation, Writing – review & editing. All authors have read and approved the final manuscript."

Use of artificial intelligence. "No generative AI tools were used in the writing of this manuscript or the analysis of its results."

References

- [1] M. Almaayah and R. Sulaiman, "Cyber risk management in the Internet of Things: Frameworks, models, and best practices," STAP Journal of Security Risk Management, vol. 1, pp. 3-23, 2024.
- [2] A. Aldossary, T. Algirim, I. Almubarak, and K. Almuhih, "Cyber security in data breaches," Journal of Cyber Security and Risk Auditing, vol. 2024, no. 1, pp. 14-22, 2024.
- [3] O. Almomani, A. M. Arabiat, A.-A. Hind, and E. Alsariera, "Hybridization of Deep Learning Models for Multiclass Attack Detection in Wireless Sensor Networks," Journal of Communications, vol. 21, no. 1, 2026.
- [4] R. K. Ayeni, A. A. Adebiyi, J. O. Okesola, and E. Igbekele, "Phishing attacks and detection techniques: A systematic review," in 2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG), 2024: IEEE, pp. 1-17.
- [5] K. Oloyede et al., "Impact Of Web (URL) Phishing and Its Detection," Valley International Journal Digital Library, pp. 484-493, 2024.

- [6] S. Al-augby, H. Alyasiri, F. G. Abdulkadhim, and Z. C. Oleiwi, "A Stacked Ensemble Classifier for Email Spam Detection via an Evolutionary Algorithm," *Mesopotamian Journal of CyberSecurity*, vol. 5, no. 2, pp. 657-670, 2025.
- [7] L. Pinjarkar, P. R. Hete, M. Mattada, S. Nejakar, P. Agrawal, and G. Kaur, "An Examination of Prevalent Online Scams: Phishing Attacks, Banking Frauds, and E-Commerce Deceptions," in *2024 Second International Conference on Advances in Information Technology (ICAIT)*, 2024, vol. 1: IEEE, pp. 1-6.
- [8] O. Almomani, A. Alsaaidah, A. A. Abu-Shareha, A. Alzaqebah, M. A. Almaiah, and Q. Shambour, "Enhance URL Defacement Attack Detection Using Particle Swarm Optimization and Machine Learning," *Journal of Computational and Cognitive Engineering*, vol. 4, no. 3, 2025.
- [9] F. M. Ghabban, "A Detection Android Cybercrime Model utilizing Machine Learning Technology," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15344-15350, 2024.
- [10] A. Kulkarni, V. Balachandran, and T. Das, "Phishing webpage detection: Unveiling the threat landscape and investigating detection techniques," *IEEE Communications Surveys Tutorials*, 2024.
- [11] O. Almomani et al., "A metaheuristic feature selection model using bat optimization for malicious URL attack detection," *Scientific Reports*, 2026, doi: <https://doi.org/10.1038/s41598-026-51981-2>.
- [12] M. Almaiah, L. Saqr, L. Al-Rawwash, L. Altellawi, R. Al-Ali, and O. Almomani, "Classification of cybersecurity threats, vulnerabilities and countermeasures in database systems," *Computers, Materials, & Continua*, vol. 81, no. 2, p. 3189, 2024.
- [13] M. H. Alsuwit, M. A. Haq, and M. A. Aleisa, "Advancing Email Spam Classification using Machine Learning and Deep Learning Techniques," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 14994-15001, 2024.
- [14] O. Almomani, A. Alsaaidah, A. A. A. Shareha, A. Alzaqebah, and M. Almomani, "Performance Evaluation of Machine Learning Classifiers for Predicting Denial-of-Service Attack in Internet of Things," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 1, 2024.
- [15] M. M. Ismail and M. A. Haq, "Enhancing enterprise financial fraud detection using machine learning," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 14854-14861, 2024.
- [16] M. K. Kumar and N. Sivakumar, "URL Phishing attack Detection using Machine Learning Algorithms," in *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*, 2024: IEEE, pp. 1-8.
- [17] L. Al-Dabbas and A. A. Abu-Shareha, "Early Detection of Female Type-2 Diabetes using Machine Learning and Oversampling Techniques," *Journal of Applied Data Sciences*, vol. 5, no. 3, pp. 1237-1245, 2024.
- [18] A. Ali, "Adaptive and context-aware authentication framework using edge AI and blockchain in future vehicular networks," *STAP Journal of Security Risk Management*, vol. 1, pp. 45-56, 2024.
- [19] Q. Qi, Z. Wang, Y. Xu, Y. Fang, and C. Wang, "Enhancing phishing email detection through ensemble learning and undersampling," *Applied Sciences*, vol. 13, no. 15, p. 8756, 2023.
- [20] S. H. Ahammad et al., "Phishing URL detection using machine learning methods," *Advances in Engineering Software*, vol. 173, p. 103288, 2022.

- [21] S. Jalil, M. Usman, and A. Fong, "Highly accurate phishing URL detection based on machine learning," *Journal of Ambient Intelligence Humanized Computing*, vol. 14, no. 7, pp. 9233-9251, 2023.
- [22] N. F. Almujaahid, M. A. Haq, and M. Alshehri, "Comparative evaluation of machine learning algorithms for phishing site detection," *PeerJ Computer Science*, vol. 10, p. e2131, 2024.
- [23] O. Aljumaiah, W. Jiang, S. R. Addula, and M. A. Almaiah, "Analyzing cybersecurity risks and threats in IT infrastructure based on NIST framework," *J. Cyber Secur. Risk Audit*, vol. 2025, no. 2, pp. 12-26, 2025.
- [24] M. Aloshoush and G. Jaradat, "Particle Swarm Optimization for Hyperparameter Tuning of One-Dimensional Convolutional Neural Networks in Wastewater Classification: An Open Problem in Feature-Dependent Generalization," *International Journal of Open Problems in Computer Science and Mathematics*, vol. 19, no. 4, pp. 15-22, 2026.
- [25] C. M. Igwilo and V. T. Odumuyiwa, "Comparative analysis of ensemble learning and non-ensemble machine learning algorithms for phishing URL detection," *FUOYE Journal of Engineering and Technology*, vol. 7, no. 3, pp. 305-312, 2022.
- [26] N. Innab et al., "Phishing Attacks Detection Using Ensemble Machine Learning Algorithms," *Computers, Materials & Continua*, vol. 80, no. 1, 2024.
- [27] R. Masadeh et al., "Narwhal Optimizer: A Nature-Inspired Optimization Algorithm for Solving Complex Optimization Problems," *Computers, Materials & Continua*, vol. 85, no. 2, pp. 3709--3737, 2025. [Online]. Available: <http://www.techscience.com/cmc/v85n2/63798>.
- [28] 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.
- [29] S. Shikalgar, S. Sawarkar, and S. Narwane, "Detection of URL based phishing attacks using machine learning," *International Journal of Engineering Research & Technology*, vol. 8, no. 11, 2019.
- [30] N. Nagy et al., "Phishing URLs detection using sequential and parallel ML techniques: comparative analysis," *Sensors*, vol. 23, no. 7, p. 3467, 2023.
- [31] 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*, vol. 11, pp. 36805-36822, 2023.
- [32] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: A machine learning approach," in *Soft computing applications in industry*: Springer, 2008, pp. 373-383.
- [33] A. A. Albishri and M. M. Dessouky, "A Comparative Analysis of Machine Learning Techniques for URL Phishing Detection," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 18495-18501, 2024.
- [34] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious urls using lexical analysis," in *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10, 2016*: Springer, pp. 467-482.
- [35] A. Arabiat and M. Altayeb, "Enhancing internet of things security: evaluating machine learning classifiers for attack prediction," *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 14, no. 5, 2024.
- [36] C. El Morr, M. Jammal, H. Ali-Hassan, and W. El-Hallak, "Data preprocessing," in *Machine Learning for Practical Decision Making: A Multidisciplinary Perspective with Applications from Healthcare, Engineering and Business Analytics*: Springer, 2022, pp. 117-163.

- [37] A. M. Arabiat, "Intelligent Model for Detecting GAN-Generated Images Based on Multi-Classifiers and Advanced Data Mining Techniques," *International Journal of Electrical and Electronic Engineering & Telecommunications*, vol. 14, no. 3, pp. 147-157, 2025.
- [38] O. Almomani et al., "A Robust Model for Android Malware Detection via ML and DL classifiers," *Mesopotamian Journal of Big Data*, vol. 2025, pp. 261-277, 2025.
- [39] A. A. Abu-Shareha and M. M. Abualhaj, "Improving Intrusion Detection System Using Feature Weighting," in *International Conference on Soft Computing and its Engineering Applications*, 2024: Springer, pp. 147-160.
- [40] A. Davarasan, J. Samual, K. Palansundram, and A. Ali, "A Comprehensive review of machine learning approaches for android malware detection," *Journal of Cyber Security and Risk Auditing*, vol. 2024, no. 1, pp. 38-60, 2024.
- [41] T. Chen et al., "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1-4, 2015.
- [42] Y. Alraba'nah, S. Al-Sharaeh, and G. Al Hindi, "Enhancing Intrusion Detection Using Hybrid Long Short-Term Memory and XGBoost," *Journal of Soft Computing and Data Mining*, vol. 6, no. 1, pp. 247-261, 2025.
- [43] V. Y. Kulkarni and P. K. Sinha, "Random forest classifiers: a survey and future research directions," *Int. J. Adv. Comput*, vol. 36, no. 1, pp. 1144-1153, 2013.
- [44] A. Alsaaidah, O. Almomani, A. A. Abu-Shareha, M. M. Abualhaj, and A. Achuthan, "ARP spoofing attack detection model in IoT network using machine learning: Complexity vs. accuracy," *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1850-1860, 2024.
- [45] M. Skurichina and R. P. Duin, "Bagging for linear classifiers," *Pattern Recognition*, vol. 31, no. 7, pp. 909-930, 1998.
- [46] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, 1997, vol. 97, pp. 211-218.
- [47] P. Bahad and P. Saxena, "Study of adaboost and gradient boosting algorithms for predictive analytics," in *International Conference on Intelligent Computing and Smart Communication 2019: Proceedings of ICSC 2019*, 2019: Springer, pp. 235-244.
- [48] M. M. Abualhaj, S. N. Al-Khatib, M. A. Alsharaiah, and M. O. Hiari, "Performance Comparison of Whale and Harris Hawks Optimizers with Network Intrusion Prevention Systems," *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1530-1538, 2024.
- [49] J. T. Hancock and T. M. Khoshgoftaar, "CatBoost for big data: an interdisciplinary review," *Journal of big data*, vol. 7, no. 1, p. 94, 2020.
- [50] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.
- [51] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 505-512.
- [52] D. S. Kapoor et al., "Comparative Evaluation and Prediction of Exoplanets Using Machine Learning Methods," *Integrating Metaheuristics in Computer Vision for Real-World Optimization Problems*, pp. 163-184, 2024.
- [53] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3-42, 2006.
- [54] M. M. Abualhaj, H. Al-Mimi, A. Al-Allawee, Q. Y. Shambour, and M. Anbar, "Enhancing Intrusion Detection Using Dragonfly Algorithm-Based Feature Selection and Extra Trees for Classification," in *2025 5th International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, 2025: IEEE, pp. 1-6.

- [55] M. M. Abualhaj et al., "Spam Detection Boosted by Firefly-Based Feature Selection and Optimized Classifiers," *Int. J. Advance Soft Compu. Appl*, vol. 17, no. 3, 2025.
- [56] D. A. Pisner and D. M. Schnyer, "Support vector machine," in *Machine learning*: Elsevier, 2020, pp. 101-121.
- [57] N. Frederick and A. Ali, "Enhancing DDoS attack detection and mitigation in SDN using advanced machine learning techniques," *Journal of Cyber Security and Risk Auditing*, vol. 2024, no. 1, pp. 23-37, 2024.
- [58] J. R. Quinlan, "Learning decision tree classifiers," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 71-72, 1996.
- [59] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 201-213, 2019.
- [60] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of translational medicine*, vol. 4, no. 11, p. 218, 2016.
- [61] A. Al Saaidah et al., "Enhancing malware detection performance: leveraging K-nearest neighbors with firefly optimization algorithm," *Multimedia Tools and Applications*, vol. 84, no. 12, pp. 10071-10094, 2025.
- [62] B. Ghogh and M. Crowley, "Linear and quadratic discriminant analysis: Tutorial," *arXiv preprint arXiv:1906.02590*, 2019.
- [63] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," *Institute for Signal and information Processing*, vol. 18, no. 1998, pp. 1-8, 1998.
- [64] M. Alazab, S. P. Rm, P. K. R. Maddikunta, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: Concepts, challenges, and future directions," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3501-3509, 2021.
- [65] F. N. Motlagh, M. Hajizadeh, M. Majd, P. Najafi, F. Cheng, and C. Meinel, "Large language models in cybersecurity: State-of-the-art," *arXiv preprint arXiv:2402.00891*, 2024.
- [66] Q. Shambour, M. Al-Zyoud, and O. Almomani, "Quantum-Inspired Hybrid Metaheuristic Feature Selection with SHAP for Optimized and Explainable Spam Detection," *Symmetry*, vol. 17, no. 10, p. 1716, 2025.