Jurnal Teknik Informatika C.I.T Medicom, 17 (6) (2026) 269-279

Published by: Institute of Computer Science (IOCS)

Jurnal Teknik Informatika C.I.T Medicom

Journal homepage: www.medikom.iocspublisher.org

Comparative Study of CatBoost, XGBoost, Random Forest, and Decision Tree for Phishing Web Page Classification

Haryani[1], Cucu Ika Agustyaningrum[2]

[1),2)] Faculty of Engineering and Informatics, Universitas Bina Sarana Informatika, Jakarta, Indonesia

Article Info

Article history

Received : Oct 23, 2025
Revised : Dec 18, 2025
Accepted : Jan 02, 2026

Keywords:

CatBoost;
Cybersecurity;
Ensemble algorithms;
Machine learning;
Phishing detection.

Abstract

Phishing is a fraudulent method in which attackers using fake websites steal user information such as login credentials and sensitive financial data. Therefore, this study compares four machine learning algorithms, namely CatBoost, XGBoost, Random Forest, and Decision Tree, in classifying phishing websites efficiently and accurately. In this study, the dataset used is the Web Page Phishing Dataset, which begins with exploration and preprocessing, which includes data cleaning, handling missing values, normalization, feature selection, and testing. Post-split. The data used has been divided into training data and test data, namely 80:20. The model was implemented using Python in Google Colaboratory. Model performance evaluation was measured in five main metrics, such as accuracy, precision, recall, F1-score, and AUC. The experimental results indicate that CatBoost achieved the best position with a performance of 89.57% in accuracy, 85.74% in F1-score, 88.73% in precision, 88.78% in recall, and 89.00% in AUC. XGBoost ranked second with a very competitive performance, followed by Random Forest, which was relatively stable with an accuracy value of 89.41% and an F1-score of 85.35%. On the other hand, the decision tree achieved the lowest performance with an accuracy of 88.69% and an F1-score of 84.10%. These performance results indicate limitations in handling complex data, as well as a tendency to overfit. Overall, ensemble boosting-based algorithms, especially CatBoost and XGBoost, outperform single trees in detecting phishing websites. These results will be benefical to progress in the next generation for the construction of intelligent based phishing detection system under machine learning. In addition, the outcomes of this study will gain momentum for future works where hyperparameter optimization, larger datasets and real-time applications for phishing detection systems can be focused. Furthermore, this work will contrast the application of ensemble algorithm in the cybersecurity field.

Corresponding Author:

Cucu Ika Agustyaningrum,
Faculty of Engineering and Informatics
Universitas Bina Sarana Informatika,
Jl. Kramat Raya No.98, RT.2/RW.9, Kwitang, Kec. Senen, Kota Jakarta Pusat, Indonesia, 10450
Email: cucu.cgy@bsi.ac.id

1. Introduction

The rapid advancement of information technology has improved convenience but also increased cybercrime, particularly phishing. Phishing uses deceptive websites and social engineering techniques

to trick users into disclosing sensitive information, posing a serious threat to personal and organizational cybersecurity [1]. Phishing websites mimic legitimate ones to deceive users into sharing confidential data, such as usernames, passwords, and PINs [2]. Phishers are people who engage in phishing; they will send emails that appear to come from banks, online services, or malicious software [3]. Phishing websites are created by unscrupulous individuals to steal personal data, so fraudsters can also disguise themselves as legitimate companies to trick consumers into visiting their websites by sending Uniform Resource Locators (URLs) to collect personal data [4]. Phishing uses fake links to threaten the privacy of individuals and companies, making it a serious cybercrime [5]. Phishing remains a significant cybersecurity threat that targets individuals and organizations by attempting to obtain information through deceptive means, such as fake websites, emails, or messages [6]. Blacklist and whitelist approaches are often used to detect phishing websites, which are based on a database of classified sites. However, this solution has limitations because not all new URLs are registered instantly. Therefore, machine learning algorithms can be used to predict phishing using data patterns, rather than explicit formulations [7].

Traditional machine learning models detect phishing but depend on manual, time-consuming URL feature extraction, making adaptation difficult as attackers create new, complex, and evolving phishing URLs [8]. Various mitigation efforts are necessary due to the increasingly significant losses caused by phishing attacks. One approach that has received considerable attention is the implementation of a phishing detection system using machine learning (ML) [9].

Several studies have been published that review phishing detection methods. Each of these papers examines recent developments from a different perspective. The authors of the journal [10], with the title "Optimization of a Web-Based Phishing Detection System Using a Decision Tree Algorithm," conducted research to develop a web-based phishing detection system utilizing the Decision Tree algorithm and the Rapid Application Development (RAD) method. The journal article [11] with the research title "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," describes an end-to-end deep neural network trained using embedded raw URLs and HTML content to detect website phishing attacks. In a paper by [12] entitled "Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision-Based Approach," they designed PishIntention as a heterogeneous deep learning vision model to address various technical challenges. This heterogeneity extracts the intent to steal credentials. The following paper, "Phishing Web Page Detection with HTML-Level Graph Neural Network" [13], The proposed model combines RNNs and GNNs, using RNNs to extract local DOM node features and GNNs to capture long-range relationships, enhancing HTML intent understanding. The paper, "Phishing Web Page Detection Methods: URL and HTML Features Detection" [14], explains test results that yield relatively similar accuracy based on the fact that URL and HTML syntax data are relatively consistent from year to year, as well as the development of rule-based applications for more effective phishing detection.

However, most studies were based primarily on algorithms or simply focused on a class of algorithmic model (e.g., URL-based models, HTML, neural networks) without providing an in-depth comparative analysis between single-tree and ensemble algorithms for the same dataset. Thus, there is still a challenge with this respect such as which algorithm to be more adaptive, stability and effective for identifying new patterns of phishing attacks.

With these similarities in mind, this study attempts to make comparisons of four machine learning algorithms (CatBoost, XGBoost, Random Forest and Decision Tree) for phishing web pages classification. The novelty of this research is the extensive measurement between the ensemble boosting and single-tree algorithms in various performance measurements so that a fairer measurement can be made regarding the advantages and disadvantages of each model. This study's outcome is anticipated to be foundational in identifying the most dependable and adaptable model for the development of a phishing detection system that is efficient, practical, and robust to contemporary digital security threats.

2. Research Methods

This study uses a quantitative approach with a comparative experimental method. The primary objective of this method is to evaluate and compare the performance of four classification algorithms CatBoost, XGBoost, Random Forest, and Decision Tree in detecting phishing web pages based on the technical features of the URL. The Web Page Phishing dataset used is obtained from an Excel file, which contains various website attributes (features) on the Kaggle page. The Web Phishing dataset comprises 20 attributes, one label, and a total dataset of 100,077 records. The following are the stages of the research method used:

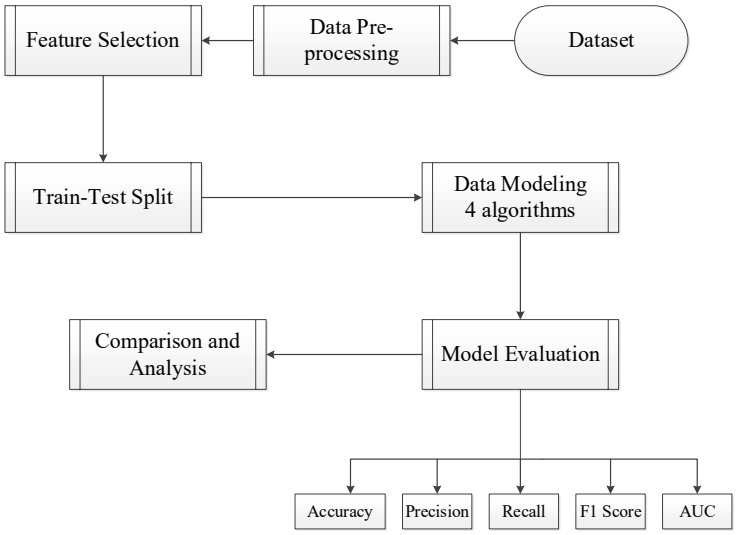

Figure 1. Research Stages

a. Dataset

This study uses secondary data taken from Kaggle (https://www.kaggle.com/datasets/danielfernandon/web-page-phishing-dataset/data). Web Phishing data has 20 attributes with one label and a total data set of 100,077. The use of classification techniques, including prediction levels and accuracy values, F1-score, precision, recall, and AUC, yields the most effective, efficient, and reliable model for use as the primary component in developing a phishing detection system that is responsive and adaptable to changes in attack patterns. But it can also be implemented practically to protect against phishing attacks on various digital platforms. In this study, a comparison was conducted between four classification algorithms: CatBoost, XGBoost, Random Forest, and Decision Tree.

Table 1. Description of attributes of the Phishing Web Page dataset

Web Page Phishing Dataset Variables					
Variable Name	**Role**	**Type**	**Description**	**Units**	**Missing Values**
url_length	Feature	Numeric	Total length of the URL	Char	No
n_dots	Feature	Numeric	Number of dots (.) in the URL	Count	No
n_hypens	Feature	Numeric	Number of hyphens (-) in the URL	Count	No
n_underline	Feature	Numeric	Number of underscores (_) in the URL	Count	No
n_slash	Feature	Numeric	Number of slashes (/) in the URL	Count	No
n_questionmark	Feature	Numeric	Number of question marks (?) in the URL	Count	No
n_equal	Feature	Numeric	Number of equal signs (=) in the URL	Count	No
n_at	Feature	Numeric	Number of at symbols (@) in the URL	Count	No
n_and	Feature	Numeric	Number of ampersands (&) in the URL	Count	No

Web Page Phishing Dataset Variables					
Variable Name	**Role**	**Type**	**Description**	**Units**	**Missing Values**
n_exclamation	Feature	Numeric	Number of exclamation marks (!) in the URL	Count	No
n_space	Feature	Numeric	Number of spaces in the URL	Count	No
n_tilde	Feature	Numeric	Number of tilde symbols (~) in the URL	Count	No
n_comma	Feature	Numeric	Number of commas (,) in the URL	Count	No
n_plus	Feature	Numeric	Number of plus signs (+) in the URL	Count	No
n_asterisk	Feature	Numeric	Number of asterisks (*) in the URL	Count	No
n_hastag	Feature	Numeric	Number of hashtags (#) in the URL	Count	No
n_dollar	Feature	Numeric	Number of dollar signs ($) in the URL	Count	No
n_percent	Feature	Numeric	Number of percent signs (%) in the URL	Count	No
n_redirection	Feature	Numeric	Number of redirection symbols (//) in the URL	Count	No
phishing	Target	Binary	Target label: 1 = phishing, 0 = legitimate	-	No

The Web Page Phishing dataset contains various features that describe the characteristics of URLs used to identify phishing patterns. The url_length variable measures the URL's length, while n_dots, n_hyphens, n_underlines, and n_slashes count specific character occurrences. Special characters such as ?, =, and @ are also recorded, as they frequently appear in phishing URLs. Additional variables, including n_space, n_plus, and n_hashtag, capture rarely used characters that may indicate suspicious activity. The n_redirection variable counts occurrences of "//", often used deceptively in fake URLs. Finally, the phishing label (1 for phishing, 0 for legitimate) serves as the target variable for machine learning classification.

b. Data Pre-Processing

In the data preprocessing research stage, 100,977 data sets were used, and the processing resulted in predictions of the most effective, efficient, and reliable models as the main components in developing a responsive and adaptive phishing detection system to changing attack patterns across various digital platforms. The data preparation process starts with selecting the relevant data, which includes examining and adjusting attribute data types. Once this selection is finalized, the following step involves cleaning the data. This is done by checking for missing values and then labeling attributes.

c. Feature Selection

The feature selection phase aims to identify the most significant attributes influencing phishing data. Once this process is completed, the data is divided into training and testing sets using a train-test split in the modeling process for the four algorithms: CatBoost, XGBoost, Random Forest, and Decision Tree.

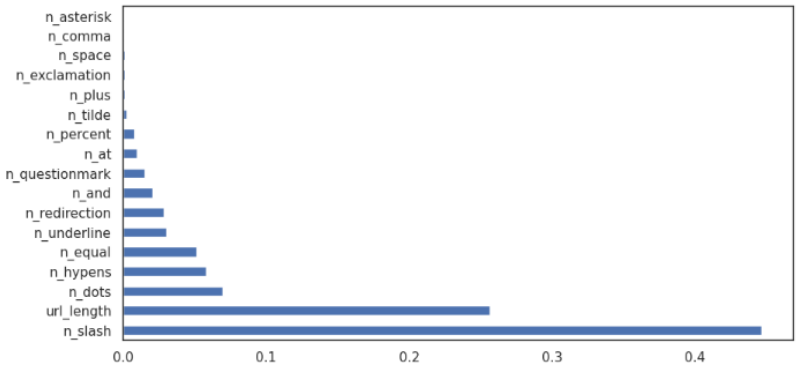

Figure 2. Features that influence phishing data

Feature importance analysis indicates that attributes such as URL length, number of special symbols, and redirection indicators contribute significantly to the model's decisions, which is consistent with the general characteristics of phishing URLs discussed in the literature.

d. Dataset Split (Train Test Split)

At this stage, the preprocessed dataset is divided into two parts: training data and testing data. The previous preprocessing process includes several essential steps, such as handling missing values and data normalization, to ensure data quality and consistency. After the data is prepared, it is divided using a train-test split method, with 80% of the data used for training the model, and the remaining 20% used for testing the model's performance. By optimizing this ratio and implementing a train-test split, this study ensures that the model can learn from sufficient data and be tested on an independent dataset to objectively measure its performance, while providing flexibility for model validation if needed[15].

e. Data Modeling Four Algorithms

Four machine-learning algorithms were employed in the modeling, including Decision Tree, Random Forest, XGBoost, and CatBoost. The Decision Tree algorithm served as a baseline model, while the Random Forest, XGBoost, and CatBoost models were used for ensemble learning. The modeling was done in the Python programming language using the Google Colaboratory platform. The algorithms were executed with primary/default parameters for this study. One of this study's constraints is the lack of hyperparameter tuning.

i) Machine Learning

Machine Learning (ML), also known as Learning Machine, is a branch of AI that focuses on learning from data, specifically developing systems that can learn "independently" without requiring repeated human programming [16]. Machine learning refers to the automated process of identifying important patterns within data. Through this approach, computers gain the ability to learn from human input and analyze data independently, without needing explicit programming. Machine learning algorithms enable systems to be trained for effective data processing [17].

ii) Python

Python is an interpreter-based programming language that supports various programming paradigms, such as object-oriented programming (OOP), functional programming, and procedural programming. In this study, Python was applied to develop a lightning analysis program on transmission lines, utilizing an object-oriented programming approach to make the code structure more modular and easier to create [18]. Python is a programming language that uses an interpreter to run program code directly without prior compilation. It is cross-platform, allowing it to run on various operating systems, including Windows, Linux, and others. Python adopts several programming paradigms from other languages, including procedural paradigms like C, object-oriented paradigms like Java, and functional paradigms like Lisp. This combination of paradigms makes Python flexible, allowing developers to build various types of projects efficiently and in a structured manner[19].

iii) CatBoost

CatBoost, or Categorical Boosting, is an implementation of Gradient Boosting that uses binary numbers from the decision tree algorithm as the basis for data prediction [20]. One of CatBoost's unique features is its gradient boosting mechanism, which is well-suited to working with heterogeneous data and can improve stability and predictive performance. This methodology employs efficient coding, which can help reduce overfitting. Such an issue can impact model accuracy, so CatBoost was created to address this shortcoming and significantly improve accuracy[21].

iv) XGBoost

XGBoost, also known as Extreme Gradient Boosting, is a highly efficient and robust machine learning algorithm designed for classification and regression tasks. It introduces several significant improvements over traditional gradient boosting implementations, including regularization to control model complexity and cache-aware optimization to improve efficiency [22]. XGBoost is a gradient boosting-based machine learning algorithm introduced by Friedman to improve performance in complex data analysis. This algorithm works by combining basis functions and weights, thus producing a model with a high degree of fit to

the data. XGBoost employs an ensemble learning approach, where multiple decision trees are constructed sequentially. Each tree created takes into account the prediction errors of the previous tree and assigns a greater weight to variables that are difficult to predict, as illustrated in Figure 3 [23].

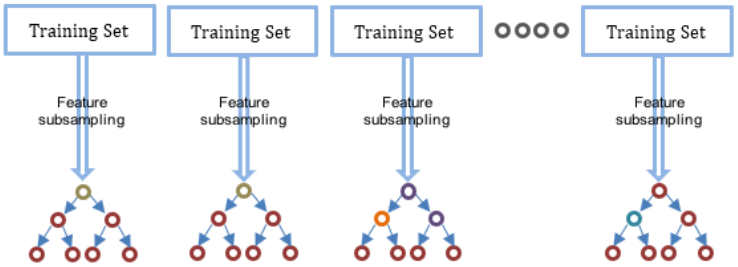

Figure 3. XGBoost Classification Representation

v) Random Forest

Random Forest is a highly versatile ensemble learning method that has found widespread application in various fields due to its robustness and high accuracy. RF excels as a classifier, particularly when learning from high-dimensional datasets or those with class imbalance, demonstrating significant success in handling complex data structures and imbalanced classification tasks [24]. Random Forests (RF) is a non-parametric ensemble method widely applied in classification and regression. It builds multiple random decision trees and ranks features by their importance to predictive outcomes [25].

vi) Decision Tree

A decision tree is a hierarchical, tree-like model consisting of a root node, internal branches, and terminal leaf nodes. Each node represents an attribute, branches represent attribute values, and leaves denote class outcomes, enabling data classification through sequential decision paths from root to leaf [26]. A decision tree predicts target values using sequential decision rules, simplifying complex relationships between inputs and outputs. It measures data uncertainty through entropy, where lower entropy indicates more homogeneous and predictable data [27]. This entropy value is then used as a basis for determining data separation at each node of the decision tree.

$$E = -\sum_{i=1}^{n} Pi\ Log_2\ Pi \tag{1}$$

$$IG\ (D, f) = E(D) - \sum_{v \in V} \left| \frac{Dv}{D} \right| E(D_v) \tag{2}$$

vii) Confusion Matrix

A confusion matrix is a useful method for examining potential bias when classifying data into different categories. It utilizes a matrix structure that represents both positive and negative classes. During model evaluation, the confusion matrix helps compute metrics such as accuracy, precision, recall, and error rate, where accuracy reflects the proportion of correctly classified instances compared to the total cases [28]. The confusion matrix can be calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

viii) Area Under Curve (AUC)
The area under the ROC curve (AUC) is a popular measure of diagnostic accuracy, estimating the probability of correctly ranking a pair of subjects. AUC is a measure of average sensitivity calculated based on all specificity values, including those that are clinically less relevant [29]. AUC is the area under the ROC curve depicted in the ROC plot. AUC is also a group-normalised average of the sensitivity, specificity, positive and negative predictive values, and positive and negative likelihood ratios [30].

ix) Model Evaluation
Model evaluation was conducted to assess the performance of four algorithms, namely CatBoost, XGBoost, Random Forest, and Decision Tree, using the Python programming language. The evaluation involves a number of key performance metrics that assess how well the model has classified the results: these include accuracy, precision, recall, F1-score, and AUC. F1-score is a holistic measure that captures both precision and recall, while accuracy reflects the proportion of all correct predictions. Precision is the measure of correct positive predictions, and recall captures the model's coverage of all positive data. On the other hand, AUC assesses how well the measures the direction of departure from the null hypothesis. AUC was selected because it gives a measurement in an overall sense about how good this model is to differentiate between phishing and legitimate classes at different decision thresholds, especially in the condition of imbalanced data. The objective of this assessment is to offer a comparison of findings as well as accuracy among all the algorithms for correctly classifying phishing data.

x) Comparison and Analysis
This study compares four machine learning algorithms CatBoost, XGBoost, Random Forest, and Decision Tree to detect phishing websites using evaluation metrics such as accuracy, recall, F1-score, and AUC. Due to having single structure, decision trees are one of the easier ML models to analyze feature relations. However, decision trees can be too complex and may lead models to overfit, and are less accurate than ensemble methods. Random forests strikes a balance between maintaining a low cost in terms of data and computation while reducing complexity and overfitting. This leads to lower overfitting in comparison to decision trees. While XGBoost requires more computational resources, it yields the best performance [31]. High accuracy, and the most consistent performance of the models, in addition to complexity, data handling, and bias from the ordered boosting [32]. Acceptable speeds and stability, with a more interpretable ensemble methods, to less outlined interpretability. XGBoost and CatBoost are the best choices. phish Detection. Decision Tree remains a good baseline, for its simplicity, and interpretability.

3. Result and Discussion

The results of the experimental study indicate that, when compared to the other algorithms, CatBoost outperformed the rest. This is based on the results of the accuracy, precision, recall, F1 measure, and AUC derived from the evaluation metrics. CatBoost's performance can be explained by several reasons. First, CatBoost handles numeric and categorical features without additional and complex encoding techniques. This results in less data loss during the pre-processing phase. Second, ordered boosting techniques used in CatBoost are essential in countering the negative of prediction bias and overfitting, particularly in the imbalanced data situations like the phishing data that need to be detected. These results provide a comprehensive overview of the effectiveness of each algorithm in detecting phishing sites and serve as the basis for a comparative analysis of model performance, as described in the next section.

a. Data Preprocessing Steps

The data preprocessing stage is a vital step before building a classification model, ensuring that the dataset is clean, consistent, and suitable for training and testing. In this study, preprocessing involved several key stages. First, data cleaning was performed to remove duplicates, incomplete entries, and irrelevant records. Then, missing values were handled using statistical imputation methods such as mean or median replacement to maintain data integrity. Next, normalization or standardization was applied to balance attribute scales, improving pattern recognition. Feature selection was then performed to identify the most relevant attributes, enhancing accuracy and reducing computational complexity. Finally, the dataset was split into 80% training and 20% testing data, preparing it for model development using CatBoost, XGBoost, Random Forest, and Decision Tree algorithms for reliable phishing detection.

b. Application of Machine Learning Algorithms

This study achieved an accuracy of 89.57% on a phishing website dataset using CatBoost, XGBoost, Random Forest, and Decision Tree algorithms. Performance was evaluated through accuracy, precision, recall, F1-score, and AUC metrics. A structured process from preprocessing to evaluation ensured reliable comparison, identifying the most effective model for phishing detection.

Table 2. Results of Phishing Web Comparison Using Machine Learning Algorithms.

Model	Accuracy	F1 Score	Precision	Recall	AUC
CatBoost	89,57%	85,74%	88,73%	88,78%	89,00%
XGBoost	89,42%	85,54%	88,58%	88,62%	89,00%
Random Forest	89,41%	85,35%	88,71%	88,36%	88,00%
Decision Tree	88,69%	84,10%	88,16%	87,25%	87,00%
Gradient Boosting	88,39%	84,25%	87,39%	87,68%	88,00%
AdaBoost	86,01%	79,77%	85,64%	83,78%	84,00%
SVM	85,78%	78,71%	86,09%	82,84%	83,00%
Logistic Regression	84,57%	76,86%	84,70%	81,51%	82,00%

This study applied several machine learning algorithms CatBoost, XGBoost, Random Forest, Decision Tree, Gradient Boosting, AdaBoost, SVM, and Logistic Regression to detect phishing websites, evaluating each using accuracy, precision, recall, F1-score, and AUC metrics. The results showed that CatBoost achieved the best performance with 89.57% accuracy and an F1-score of 85.74%, followed closely by XGBoost (89.42% accuracy) and Random Forest (89.41% accuracy). The Decision Tree performed slightly lower due to its simpler structure and tendency to overfit. Gradient Boosting and AdaBoost achieved moderate results, while SVM and Logistic Regression showed lower accuracy and recall, indicating limitations in handling complex phishing patterns. Overall, the comparison confirmed that ensemble boosting algorithms especially CatBoost and XGBoost outperform other models in accuracy and consistency, making them the most suitable for phishing detection systems.

c. Model Comparison

The comparison chart illustrates the performance of eight machine learning algorithms in detecting phishing websites, evaluated using five metrics: accuracy, F1 score, precision, recall, and AUC.

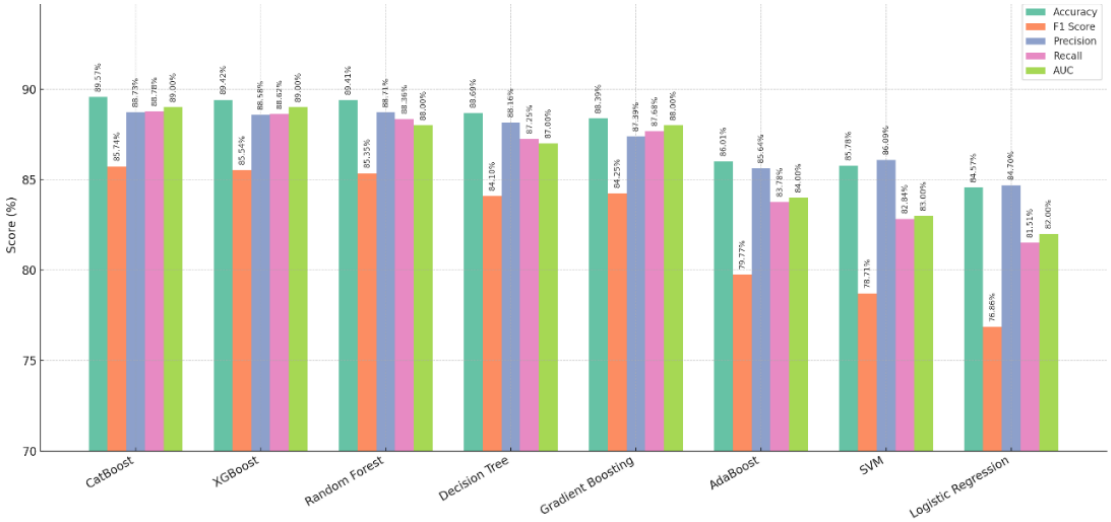

Figure 4. Comparison Chart of Algorithm Calculation Results

Overall, the CatBoost algorithm demonstrated the best performance, achieving the highest accuracy of 89.57%, an F1-score of 85.74%, and a strong balance between precision (88.73%) and recall (88.78%), indicating its reliability and consistency in detecting phishing websites. The XGBoost model followed closely, with 89.42% accuracy, an F1-score of 85.54%, and an AUC of 89.00%, making it a strong and efficient alternative to CatBoost. Random Forest also performed well, achieving 89.41% accuracy and an F1-score of 85.35%, which offered stability but slightly lower performance compared to the boosting algorithms. In contrast, the Decision Tree achieved an accuracy of 88.69% and an F1-score of 84.10%. Although interpretable and straightforward, it lagged behind ensemble models due to its tendency to overfit. Gradient Boosting produced moderate results (88.39% accuracy, 84.25% F1-score), performing adequately but not matching CatBoost or XGBoost. Meanwhile, AdaBoost and SVM underperformed, with AdaBoost achieving 86.01% accuracy and an F1-score of 79.77%, and SVM attaining 85.78% accuracy, demonstrating good precision (86.09%) but lower recall (82.84%). Lastly, Logistic Regression ranked lowest, with 84.57% accuracy and an F1-score of 76.86%, indicating that it is less effective for complex phishing detection compared to advanced ensemble algorithms.

The current results are consistent with previously discussed findings (intro); CatBoost and XGBoost were found to be the most suitable for complex and nonlinear data in the cybersecurity classification problems. XGBoost showed a very competitive performance with a short training time and efficient regularization. On the other hand, Random Forest had consistent performance, although it was more expensive to compute. The decision tree is interpretable, but its lower accuracy and F1 score showed it struggling with the sophistication of the phishing URL data. It should be noted that the performance gap between CatBoost and XGBoost is small (approximately between 0.1 and 0.2 percent). Since the difference has not been statistically tested, there is no way to know if it is significant or not. Thus, the results should be understood to indicate relative performance and not indicate that there is absolute superiority. To solidify the results, it would be beneficial for future studies to perform statistical significance testing and analysis on larger and more varied datasets. From a cybersecurity standpoint, the results show that, particularly CatBoost and XGBoost, boosting-based ensemble algorithms have promising applications for adaptive and trustworthy phishing detection systems.

4. Conclusions

This research analyzed four algorithms to classify phishing websites: CatBoost, XGBoost, Random Forest, and Decision Tree, based on each algorithm's experimental results and performance. CatBoost

demonstrated the highest performance and achieved the highest accuracy at 89.57%, followed by XGBoost and Random Forest, leading to a stable performance comparison. In contrast, due to the tendency to overfit and the inability to manage complicated data structures, the Decision Tree demonstrated the lowest performance. CatBoost and XGBoost, being boosting-based ensemble algorithms, excelled compared to the single-tree algorithms for detecting phishing sites. This study empirically proves boosting algorithms' empirical phishing detection URL-based data literature to practically assist structured machine learning phishing detection systems. Despite its promising results, this study has several limitations, including the use of a dataset that only includes URL features without considering HTML content, DOM structure, or visual characteristics of web pages. Furthermore, hyperparameter tuning and statistical significance testing have not been performed, so performance differences between models cannot yet be declared mathematically significant. Therefore, further research is recommended to use larger and more diverse datasets, integrate HTML and visual features, perform hyperparameter optimization, and test the model in a real-time environment. Exploring other algorithms such as LightGBM or deep learning approaches also has the potential to improve the effectiveness of future phishing detection systems.

Acknowledgement

Gratitude is expressed to the Bina Sarana Informatika Foundation for providing support in the form of research funding and community service for lecturers through foundation/internal funds, which are managed by the Institute for Research and Community Service (LPPM) of Bina Sarana Informatika University. This research was carried out with the support of internal grant funds from the Bina Sarana Informatika Foundation for the 2025 fiscal year (Number: 108/LPPM-UBSI/VII/2025) with the title "Determination of the Winners of the Internal Grant Proposal for Research and Community Service Funding from the Bina Sarana Informatika Foundation in 2025."

References

[1] R. Zieni, L. Massari, and M. C. Calzarossa, "Phishing or Not Phishing? A Survey on the Detection of Phishing Websites," *IEEE Access*, vol. 11, no. February, pp. 18499–18519, 2023, doi: 10.1109/ACCESS.2023.3247135.

[2] A. F. Mahmud and S. Wirawan, "Deteksi Phishing Website menggunakan Machine Learning Metode Klasifikasi," *Sist. J. Sist. Inf.*, vol. 13, no. 4, pp. 2540–9719, 2024, doi: 10.32520/stmsi.v13i4.

[3] V. A. Windarni *et al.*, "Deteksi Website Phishing Menggunakan Teknik Filter Pada Model Machine Learning," *Inf. Syst. J.*, vol. 6, no. 1, pp. 39–43, 2023, doi: 10.24076/infosjournal.2023v6i01.

[4] Y. Miftahuddin and M. M. Faturrahman, "Penerapan Data Standardization dan Multilayer Perceptron pada Identifikasi Website Phishing," *J. MIND J. | ISSN*, vol. 7, no. 2, pp. 111–123, 2022, doi: 10.26760/mindjournal.v7i2.111-123.

[5] P. Subarkah and A. N. Ikhsan, "Identifikasi Website Phishing Menggunakan Algoritma Classification And Regression Trees (CART)," *J. Ilm. Inform.*, vol. 6, no. 2, pp. 127–136, 2021, doi: 10.35316/jimi.v6i2.1342.

[6] A. Fajar, S. Yazid, and I. Budi, "Enhancing Phishing Detection through Feature Importance Analysis and Explainable AI: A Comparative Study of CatBoost, XGBoost, and EBM Models," 2024, doi: 10.48550/arXiv.2411.06860.

[7] R. Saputra and E. Hartati, "Deteksi Website Phising Menggunakan Algoritma Random Forest Dengan Optimalisasi Gridsearch," *JUTIM (Jurnal Tek. Inform. Musirawas)*, vol. 10, no. 1, pp. 55–67, 2025, doi: 10.32767/jutim.v10i1.2674.

[8] S. Asiri, Y. Xiao, S. Alzahrani, S. Li, and T. Li, "A Survey of Intelligent Detection Designs of HTML URL Phishing Attacks," *IEEE Access*, vol. 11, no. January, pp. 6421–6443, 2023, doi: 10.1109/ACCESS.2023.3237798.

[9] Lukito and W. B. T. Handaya, "Deteksi Website Phishing Menggunakan Teknik Machine Learning," *J. Inform. Atma Jogja*, vol. 6, no. 01, pp. 69–80, 2025, doi: 10.24002/jiaj.v6i1.11538.

[10] M. R. Fatiha, I. Setiawan, A. N. Ikhsan, and I. R. Yunita, "Optimisasi Sistem Deteksi Phishing Berbasis Web Menggunakan Algoritma Decision Tree," *J. Ilm. IT CIDA*, vol. 10, no. 2, p. 97, 2024, doi: 10.55635/jic.v10i2.212.

[11] C. Opara, Y. Chen, and B. Wei, "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," *Expert Syst. Appl.*, vol. 236, no. October 2020, p. 121183, 2024, doi: 10.1016/j.eswa.2023.121183.

[12] R. Liu, Y. Lin, X. Yang, S. H. Ng, D. M. Divakaran, and J. S. Dong, "Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision Based Approach," *Proc. 31st USENIX Secur. Symp. Secur. 2022*, pp. 1633–1650, 2022.

[13] L. Ouyang and Y. Zhang, "Phishing Web Page Detection with HTML-Level Graph Neural Network," *Proc. - 2021 IEEE 20th Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2021*, pp. 952–958, 2021, doi: 10.1109/TrustCom53373.2021.00133.
[14] H. Faris and S. Yazid, "Phishing Web Page Detection Methods: URL and HTML Features Detection," *IoTaIS 2020 - Proc. 2020 IEEE Int. Conf. Internet Things Intell. Syst.*, pp. 167–171, 2021, doi: 10.1109/IoTaIS50849.2021.9359694.
[15] A. Rácz, D. Bajusz, and K. Héberger, "Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification," *Molecules*, vol. 26, no. 4, p. 1111, Feb. 2021, doi: 10.3390/molecules26041111.
[16] I. H. Sarker, "Machine Learning : Algorithms , Real - World Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 3, pp. 1–21, 2021, doi: 10.1007/s42979-021-00592-x.
[17] C. I. Agustyaningrum, Y. Ramdhani, D. Purnama Alamsyah, and O. I. B. Hariyanto, "Deep neural networks and conventional machine learning classifiers to analyze thoracic survival data," *IAES Int. J. Artif. Intell.*, vol. 13, no. 3, pp. 3686–3694, 2024, doi: 10.11591/ijai.v13.i3.pp3686-3694.
[18] V. Cutting and N. Stephen, "A Review on using Python as a Preferred Programming Language for Beginners," *Int. Res. J. Eng. Technol.*, vol. 08 Issue:, no. 08, pp. 4258–4263.
[19] M. A. Hamid, D. Aditama, E. Permata, N. Kholifah, M. Nurtanto, and N. W. A. Majid, "Simulating the COVID-19 epidemic event and its prevention measures using python programming," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 26, no. 1, pp. 278–288, 2022, doi: 10.11591/ijeecs.v26.i1.pp278-288.
[20] J. Qi, R. Yang, and P. Wang, "Application of explainable machine learning based on Catboost in credit scoring," *Electrochem. Soc. J. Phys. Conf. Ser.*, p. https://iopscience.iop.org/article/10.1088/1742-65, 2021, doi: 10.1088/1742-6596/1955/1/012039.
[21] D. Wang and H. Qian, "CatBoost-Based Automatic Classification Study of River Network," *ISPRS Int. J. Geo-Information*, vol. 12, no. 10, p. 416, Oct. 2023, doi: 10.3390/ijgi12100416.
[22] F. Yi *et al.*, "XGBoost-SHAP-based interpretable diagnostic framework for alzheimer's disease," *BMC Med. Inform. Decis. Mak.*, vol. 23, no. 1, p. 137, Jul. 2023, doi: 10.1186/s12911-023-02238-9.
[23] Hanif Abdul Karim Afandi, M. Lazaro Fa. Al-Dzaki, Nurul Qomariasih, and Reza Aulia Wildana, "GuardSurfing : Ekstensi Browser sebagai Alat Bantu Deteksi Website Phishing dengan Metode Klasifikasi XGBoost untuk Deteksi URL Phishing Berbasis Flask Framework," *Info Kripto*, vol. 19, no. 2, pp. 73–85, Sep. 2025, doi: 10.56706/ik.v19i2.124.
[24] K. Kirso and M. D. Anasanti, "Improving Alzheimer's Disease Prediction Accuracy using Feature Selection, K Fold Cross Validation, and KNN Imputer Techniques," *Telematika*, vol. 18, no. 1, pp. 75–90, 2025, doi: 10.35671/telematika.v18i1.3055.
[25] M. C. E. Simsekler, N. H. Alhashmi, E. Azar, N. King, R. A. M. A. Luqman, and A. Al Mulla, "Exploring drivers of patient satisfaction using a random forest algorithm," *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, p. 157, Dec. 2021, doi: 10.1186/s12911-021-01519-5.
[26] M. R. Maskur A and A. Wibowo, "Taxpayer Awareness Classification Using Decision Tree and Naïve Bayes Methods," *J. Appl. Informatics Comput.*, vol. 8, no. 1, pp. 47–54, 2024, doi: 10.30871/jaic.v8i1.6654.
[27] H. Syahputra, S. I. Naibaho, M. A. Maulana, I. Zulfahmi, and E. P. Sinaga, "Perbandingan Algoritma Support Vector Machine (SVM) dan Decision Tree Untuk Deteksi Tingkat Depresi Mahasiswa," *Bina Insa. Ict J.*, vol. 10, no. 1, p. 52, 2023, doi: 10.51211/biict.v10i1.2304.
[28] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-Label Confusion Matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022, doi: 10.1109/ACCESS.2022.3151048.
[29] S. Parodi, D. Verda, F. Bagnasco, and M. Muselli, "The clinical meaning of the area under a receiver operating characteristic curve for the evaluation of the performance of disease markers," *Epidemiol. Health*, vol. 44, pp. 1–10, 2022, doi: 10.4178/epih.e2022088.
[30] A. M. Carrington *et al.*, "Deep ROC Analysis and AUC as Balanced Average Accuracy, for Improved Classifier Selection, Audit and Explanation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 329–341, 2023, doi: 10.1109/TPAMI.2022.3145392.
[31] H. E. Sadig *et al.*, "Advanced time complexity analysis for real-time COVID-19 prediction in Saudi Arabia using LightGBM and XGBoost," *J. Radiat. Res. Appl. Sci.*, vol. 18, no. 2, p. 101364, Jun. 2025, doi: 10.1016/j.jrras.2025.101364.
[32] Y. Cao, T. E. Schartel, D. C. Houghton, J. Ross, and D. M. Infante, "Ecological Informatics Evaluating machine leaning algorithms for accuracy , stability , and among-predictors discriminability in modeling species-richness across ten datasets," *Ecol. Inform.*, vol. 90, no. January, p. 103323, 2025, doi: 10.1016/j.ecoinf.2025.103323.