INTENSIF, Vol.10 No.2 2026
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Optimization of Boosting-based Classification for Phishing Web Detection Using Ant Colony Optimization

Received:
10 May 2026
Accepted:
17 July 2026
Published:
22 August 2026

[1*]Kadek Adies Wiranegara, [2]Dandy Pramana Hostiadi, [3]Roy Rudolf Huizen
[1]Magister Program, Department of Magister Information Systems, Institut Teknologi dan Bisnis STIKOM Bali, Indonesia
[2,3]Department of Magister Information Systems, Institut Teknologi dan Bisnis STIKOM Bali, Indonesia
E-mail: [1]232012015@stikom-bali.ac.id, [2]dandy@stikom-bali.ac.id, [3]roy@stikom-bali.ac.id

*Corresponding Author

Abstract— **Background:** Cybercriminals commonly use phishing attacks by manipulating domain and website characteristics to mislead users into revealing sensitive personal information. The increasing scale of phishing attacks demands automated detection mechanisms that are accurate, efficient, and reproducible. **Objective**: The purpose of this research is to compare and evaluate the performance of boosting-based machine learning models for phishing domain detection integrated with Ant Colony Optimization (ACO) for feature selection. This study also aims to analyze the impact of ACO-based feature selection on classification performance and feature efficiency under consistent experimental conditions. **Methods**: Experiments were conducted using a public phishing webpage dataset from Kaggle, comprising 11,430 samples and 87 numerical features extracted from URL structures and webpage characteristics. Two scenarios were evaluated: training models with all features and with a reduced feature subset selected by ACO. Performance was assessed using Accuracy, Precision, Recall, F1-score, and ROC–AUC under a fixed train–test split and predefined hyperparameters. **Result**: Experimental results showed that LightGBM without feature selection achieved the highest accuracy 97.24%. However, ACO reduced feature dimensionality and improved computational efficiency in some models, including faster execution and lower memory usage for LightGBM, while also slightly decreasing accuracy. These findings indicate that ACO effectiveness is model-dependent and involves a balance between predictive performance and computational efficiency. **Conclusion**: The results confirm that boosting-based models effectively detect phishing domains, while ACO showed model-dependent effects on feature efficiency and computational trade-offs. Future work should use diverse datasets and systematic hyperparameter optimization to improve generalizability and performance.

Keywords— Phishing Detection; Boosting Algorithms; Ant Colony Optimization; Feature Selection; Machine Learning

Corresponding Author:

Kadek Adies Wiranegara,
Department of Magister Information Systems,
Institut Teknologi dan Bisnis STIKOM Bali,
Email: 232012015@stikom-bali.ac.id
Orchid ID: http://orcid.org/ 0009-0000-7199-2332

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

I. INTRODUCTION

Phishing is one of the most persistent and adaptive forms of cybercrime because it exploits the human factor as its primary vulnerability [1]. Many victims are unaware that they are being targeted or tend to perceive security warnings as trivial. In general, these attacks tend to target two groups of users: (1) new users who lack an understanding of the technical aspects of the internet and information security, and (2) users who actually understand the risks but behave carelessly, thereby ignoring attack indicators that appear in their daily digital activities. This condition makes phishing a threat that is not only technical but also social, as its success depends heavily on user behavior and decision-making [2].

In practice, phishing actors typically construct fraudulent websites that mimic legitimate services, then distribute links via email, social media, messaging applications, or other electronic communication channels [3]. These links often appear legitimate because attackers impersonate trusted entities (brand impersonation) and design login pages that closely resemble the original ones. When victims enter their credentials (username and password), attackers capture the data and can use it for account takeover, identity theft, data breaches, and financial losses. The impact is not only experienced by individuals but also by organizations, as phishing incidents can lead to system compromise, reputational damage, and high recovery costs [4].

The scale of this threat continues to grow alongside the expansion of digital infrastructure and the ease of domain registration. Cybersecurity industry reports indicate a significant growth trend in phishing domains over certain periods, including an increase in newly registered domains used by phishers [5]. The rising volume and diversity of these domains reinforce the need for automated detection mechanisms that can operate quickly and accurately on large-scale data, as manual approaches are no longer cost- or time-efficient [6].

To address this issue, various studies have proposed machine learning–based approaches to phishing detection that leverage URL-based characteristics and feature-engineered representations. Previous research explored the use of deterministic and probabilistic neural network models as an approach for phishing URL detection, where the probabilistic model demonstrated superior performance by providing uncertainty estimates in its predictions, thereby improving interpretability in security-related decision-making [7]. However, the study primarily focused on probabilistic neural network behavior and uncertainty estimation, while comparative investigations of boosting-based classifiers and metaheuristic feature optimization techniques were not discussed in depth. The difference between this research and previous research is that this study integrates boosting-based classification algorithms with Ant Colony Optimization

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

(ACO) to systematically evaluate the contribution of feature selection toward phishing detection performance under reproducible experimental settings.

Prior studies investigated phishing URL detection by combining different machine learning approaches, including XGBoost and Categorical Boosting, with feature selection strategies such as SHAP and Recursive Feature Elimination (RFE) [8]. The study reported promising classification accuracy and a competitive AUC, indicating that boosting algorithms are highly effective for phishing detection tasks. However, the study primarily emphasized improvements in predictive performance and did not provide a comprehensive comparison of multiple boosting algorithms under identical preprocessing procedures and evaluation protocols. The difference between this research and previous research is that this study systematically compares several boosting-based classifiers, including AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost, using consistent experimental configurations to provide a more comprehensive analysis of model behavior, classification stability, and computational efficiency.

Earlier research investigated phishing detection by combining Random Forest and Artificial Neural Network methods with dimensionality reduction and feature selection techniques, including PCA and RFE [9]. Experimental findings demonstrated that feature selection techniques reduced overfitting while improving computational efficiency. However, the study primarily evaluated conventional feature selection approaches and did not extensively investigate metaheuristic-based optimization strategies such as Ant Colony Optimization for feature subset selection. The difference between this research and previous research is that this study applies ACO-based feature optimization across multiple boosting algorithms to analyze how optimized feature subsets affect phishing classification accuracy, AUC, model robustness, and efficiency.

ACO was adopted in this study because selecting relevant features for phishing detection requires solving a combinatorial optimization problem, where the objective is to identify the most informative subset of features from a high-dimensional feature space. Previous studies have shown that ACO is particularly effective for feature subset selection because its pheromone-guided probabilistic search mechanism enables adaptive exploration of feature combinations while simultaneously balancing exploration and exploitation during optimization [10], [11]. Unlike conventional wrapper methods such as Recursive Feature Elimination (RFE), which iteratively remove features based on ranking criteria, ACO searches for collective feature subsets and can better capture feature interactions and redundancy patterns. In addition, compared with other swarm-based optimization methods such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), ACO has demonstrated advantages in discrete combinatorial optimization problems because its pheromone updating mechanism preserves collective learning

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

from previously successful feature subsets and improves adaptive search behavior in large search spaces [11].

Despite the promising performance achieved in previous studies, several methodological challenges remain unresolved in phishing detection research. Modern phishing datasets frequently contain high-dimensional features with varying levels of relevance, where redundant or weakly informative features may negatively affect model generalization, increase computational complexity, and introduce overfitting [12], [13]. In addition, many existing studies primarily focus on achieving high predictive accuracy without thoroughly examining the consistency of evaluation procedures, the reproducibility of experimental settings, and the comparative behavior of multiple boosting-based classifiers. Furthermore, few studies have comprehensively investigated the integration of metaheuristic optimization techniques with boosting algorithms for phishing detection, particularly regarding feature reduction and model efficiency.

Based on these limitations, the purpose of this research is to systematically evaluate and optimize boosting-based phishing detection models by integrating Ant Colony Optimization (ACO) as a feature selection strategy within reproducible experimental settings. This study emphasizes two primary aspects: (1) comparative evaluation of several boosting-based machine learning algorithms, and (2) optimization of feature subsets to reduce feature complexity while maintaining classification effectiveness. The research uses a publicly available phishing dataset from Kaggle that includes URL-related numerical attributes and phishing indicators. After preprocessing, including the removal of irrelevant string-based attributes and label separation, the dataset is used to train multiple boosting classifiers with both full-feature and optimized-feature configurations. Through this experimental design, the study aims not only to achieve high classification accuracy but also to identify feature subsets that significantly contribute to distinguishing phishing domains from legitimate websites.

The novelties of this study are as follows: (i) proposing an optimized phishing domain detection framework by integrating Ant Colony Optimization (ACO) as a feature selection strategy within boosting-based classifiers, enabling a systematic evaluation of how feature optimization influences different boosting learners; (ii) providing an in-depth experimental analysis of multiple boosting-based classifiers, including AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost, under consistent and reproducible experimental settings; and (iii) empirically demonstrating the model-dependent impact of ACO-based feature optimization on classification performance, highlighting scenarios where boosting classifiers benefit from reduced and more informative feature subsets. Accordingly, this research provides new insights into the effective combination of metaheuristic feature selection and boosting-based

INTENSIF, Vol.10 No.2 2026
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

machine learning for phishing detection, while offering a practical, reproducible approach that balances detection accuracy, model stability, and computational efficiency.

II. RESEARCH METHOD

This section describes the dataset, the overall research model flow, the preprocessing procedures, and the experimental setup used in this study. As illustrated in Figure 1, the proposed framework begins with a public dataset of phishing webpages that includes numerical features extracted from URL structures, webpage characteristics, and domain-related attributes. During preprocessing, non-informative attributes, such as raw URL strings, are removed because phishing detection models generally rely on feature-based numerical representations rather than raw textual URLs for classification [14]. The categorical target labels are then transformed into binary numerical class representations (0 for legitimate and 1 for phishing) to ensure compatibility with supervised machine learning algorithms [15]. To preserve class distribution across subsets, the dataset was divided using a fixed stratified train–test split, which also contributed to reproducibility and equitable comparison among models [16]. The novelty of this proposed model is indicated by the red borderline.

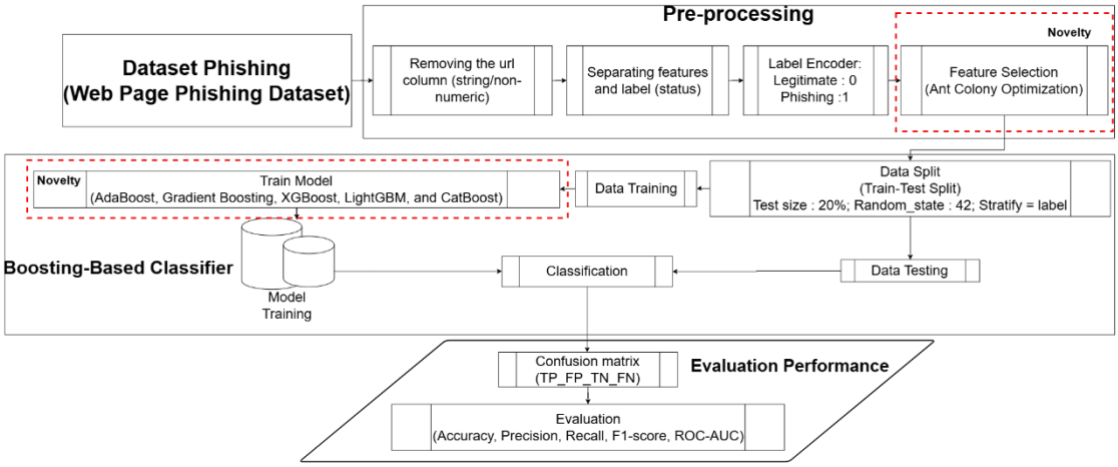

Fig 1. Overall Experimental Framework for Phishing Domain Detection using Boosting-Based Classifiers and ACO-Based Feature Selection

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

A. Research Design Experimental

This study applies a quantitative experimental design to evaluate boosting-based machine learning methods for phishing domain detection in a controlled, comparable setup. The experiments are arranged into two scenarios to assess the impact of feature selection:

1. Scenario A (Primary): Without feature selection, models are trained on all available extracted features, serving as a baseline for the best achievable performance when no feature reduction is applied.
2. Scenario B (Comparative/Ablation): With feature selection, Ant Colony Optimization (ACO) is applied to the training set to select a reduced feature subset before model training, enabling analysis of whether feature reduction improves performance and/or efficiency.

To ensure fairness, both scenarios use the same dataset and a consistent train-test partition, so performance differences reflect methodological differences rather than data variation.

B. Pseudocode of the Proposed Method

Algorithm 1 presents the pseudocode for classification without feature selection, while Algorithm 2 describes the classification process with ACO-based feature selection.

Table 1. Pseudocode of the Proposed Method

Algorithm 1. Pseudocode without Feature Selection	Algorithm 2. Pseudocode with ACO-Based Feature Selection
Inputs: 1. Dataset (X, y) 2. Train–test split ratio 3. Boosting classifiers (AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost)	**Inputs:** 1. Dataset X, y 2. Number of ants 3. Number of iterations 4. Cross-validation folds Boosting classifiers (AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost)
Output: 1. Predicted class labels 2. Performance metrics (Accuracy, Precision, Recall, F1-score, ROC–AUC)	**Output:** 1. Selected feature subset 2. Predicted class labels Performance metrics (Accuracy, Precision, Recall, F1-score, ROC–AUC)
Process: 1. Load the dataset containing extracted phishing-related features and class labels. 2. Separate the dataset into a feature matrix and target labels. 3. Split the dataset into training and testing sets using a fixed random seed. 4. Initialize the classification models with predefined hyperparameters. 5. Train each boosting-based classifier using the training data and the full feature set.	**Process:** 1. Load the dataset and separate features $X$and labels y 2. Initialize pheromone values for all features. 3. For each iteration: -For each ant: -Candidate feature subsets are generated using pheromone and heuristic information.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

6. Generate predictions on the testing data for each model. 7. Compute performance metrics based on the predicted and true labels. 8. Report the classification results for all models.	-Evaluate the candidate subset using cross-validation on the training data. -Update pheromone values according to the evaluation results. 4. Select the feature subset with the highest evaluation score as the final selected features. 5. Generate predictions for all models on the test data. 6. Split the reduced dataset into training and testing sets using the same fixed random seed. 7. Train each boosting-based classifier using the selected features. 8. Generate predictions on the test data for all models. 9. Compute evaluation metrics based on the predicted labels and the ground truth labels. 10. Report classification results and the selected feature subset.

C. Dataset and Feature Representation

The dataset used is a public phishing webpage dataset obtained from Kaggle (https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset), containing 11,430 instances and 89 total attributes. The dataset consists of phishing and legitimate URLs (secondary data). In the journal and analysis, the learning features are treated as 87 extracted (engineered) features, which is consistent if the 89 columns include:

1. URL (string / non-informative raw text)
2. Label/status
3. 87 numeric extracted features used for modeling.

The class distribution in the report is described as balanced (phishing vs legitimate).

Table 2. Dataset Description

Url	Lenght_Url	Length_Hostname	Ip	….	Status
http://www.crestonwood.com/router.php	37	19	0	…	legitimate
http://rgipt.ac.in	18	11	0	….	legitimate
http://www.mutuo.it	19	12	0	…	phishing
https://www.chiefarchitect.com/	31	22	0	….	phishing

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

D. Data Preprocessing

Data preprocessing is applied to enhance consistency and ensure that the dataset is appropriate for boosting-based learning methods:

1. Remove non-informative attribute(s): raw URL strings are excluded from training because the models are trained on extracted numerical features.
2. Label encoding: class labels (phishing/legitimate) are encoded into numerical form for binary classification.
3. Data quality checks: the report indicates that the dataset is already clean and can be used directly for modeling (no major missing/invalid values).
4. Consistent split: the same train–test split is used across all models and both scenarios to ensure comparability [17],[18].

E. Boosting-Based Classification Models

In this study, five boosting models are evaluated for binary classification (phishing vs. legitimate), namely AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost. A consistent training and testing dataset is applied to all models to ensure that performance comparisons are conducted fairly. Boosting models construct the prediction function additively, with new models added to correct the errors of previous models [19]. In general, an ensemble predictor can be expressed as (Equation 1):

$$f_m(x) = \sum_{m=1}^{M} v f_m(x) \qquad (1)$$

where $f_m(x)$ fdenotes the weak learner at the -m iteration, and ν is the learning rate.

1. AdaBoost

AdaBoost constructs a sequence of weak classifiers by adjusting sample weights [20]. Samples that are misclassified in the previous iteration are assigned higher weights, so subsequent models focus more on these errors, as summarized in Equations 2 to 6.

Initial sample weights:

$$w_i^{(1)} = \frac{1}{N} \qquad (2)$$

weighted error at iteration *t*:

$$\varepsilon_t = \sum_{i=1}^{N} w_i^{(t)} I\left(y_i \neq h_t(x_i)\right), \qquad (3)$$

INTENSIF, Vol.10 No.2 2026
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

weight of the weak learner:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \quad (4)$$

Sample weight update (for $yi \in \{-1, +1\}$):

$$w_i^{(t+1)} = w_i^{(t)} exp\left(-\alpha_t y_i h_t(x_i)\right) \quad (5)$$

Final prediction:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right) \quad (6)$$

2. Gradient Boosting

The Gradient Boosting approach optimizes the loss function incrementally at each stage by relying on negative gradients, often referred to as pseudo-residuals. In the context of binary classification, log-loss is widely used as the objective function [21], as described in Equations 7 to 10.

$$L\left(y, F(x)\right) = log\left(1 + exp\left(-yF(x)\right)\right), y \in -1, +1 \quad (7)$$

pseudo-residual at iteration m:

$$r_{im} = -\left[\frac{\partial L\left(y_i, F(x_i)\right)}{\partial F(x_i)}\right] F = F_{m-1,} \quad (8)$$

the weak learner $h_m(x)$ is trained to approximate the residual:

$$h_m = arg\ min_h \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (r_{im} - h(x_i))^2, \quad (9)$$

Update model:

$$F_m(x) = F_m - 1(x) + \nu\gamma_m h_m(x) \quad (10)$$

3. XGBoost

XGBoost is an extension of Gradient Boosting that incorporates regularization and employs second-order (Newton) optimization to improve stability and efficiency. In this implementation, training is evaluated using log-loss (eval_metric = "logloss"), meaning the

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

loss function is the log-loss for binary classification [22], as summarized in Equations 11 to 14.

Additive model:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), \tag{11}$$

regularized objective:

$$J = \sum_{i=1}^{N} \ell(y_i, \hat{y}_i) + \sum_{i=1}^{T} \Omega(f_t) \tag{12}$$

optimal leaf weight:

$$w_j^* = -\frac{\sum_{i \in I_j} gi}{\sum_{i \in I_j} hi + \lambda} \tag{13}$$

gain split:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} + \frac{G^2}{H + \lambda}\right] - \gamma \tag{14}$$

4. LightGBM

LightGBM is an optimized implementation of GBDT designed for efficiency through histogram-based splitting and a leaf-wise tree growth strategy. For binary classification, the commonly used default objective is the binary log-loss [23]**.** In general, LightGBM still employs gradient-based boosting formulations, so its update form is equivalent to Equations 15 and 16**:**

$$J^{(t)} \approx \sum_{i=1}^{N}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2\right] + \Omega(f_t) \tag{15}$$

and the optimal leaf weight:

$$w_j^* = \frac{\sum_{i \in I_j} gi}{\sum_{i \in I_j} hi + \lambda} \tag{16}$$

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

The main difference of LightGBM lies in its computational strategy (histogram binning) and the selection of the leaf with the largest gain for splitting, resulting in faster training and often yielding high performance on large-scale datasets [24].

5. CatBoost

CatBoost is a gradient boosting method designed to improve generalization and reduce training bias (prediction shift), particularly when the dataset contains categorical features [25]. Although the main features in this study are numerical (since string-based URL columns are removed), CatBoost is still evaluated for its stability in decision-tree–based boosting. CatBoost employs more robust boosting strategies (ordered boosting) to reduce residual estimation bias, thereby potentially providing stable performance across diverse data characteristics. For binary classification, the commonly used loss function is log-loss, as described in Equation 17**:**

$$\ell\big(y, F(x)\big) = log\left(1 + exp\big(-yF(x)\big)\right) \qquad (17)$$

F. Hyperparameter Policy

To maintain fairness in algorithm comparison and ensure reproducibility of the results, this study adopts a fixed/manual hyperparameter policy. This means that each model is trained using a predefined parameter configuration. This approach is chosen, so any observed performance differences more accurately reflect the intrinsic characteristics of each boosting algorithm, rather than differences arising from the extent of hyperparameter tuning. Besides, all models use the same train–test split and the same random seed (seed = 42) to reduce variability due to stochastic processes [26].

Table 3. Configured Hyperparameters

Model	Configured hyperparameters	Value
LightGBM	random_state, verbose	42, -1
CatBoost	iterations	300
	learning_rate	0.1
	depth	6
	random_seed	42
	verbose	False
AdaBoost	n_estimators	300
	learning_rate	0.1
	random_state	42
Gradient Boosting	n_estimators	300
	learning_rate	0.1
	random_state	42
XGBoost	n_estimators	300
	learning_rate	0.1

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Model	Configured hyperparameters	Value
	max_depth	6
	subsample	0.8
	colsample_bytree	0.8
	random_state	42
	eval_metric	logloss

G. ACO Parameters & Objective Function

Feature selection is performed using an Ant Colony Optimization (ACO) wrapper before training the boosting models [27]. ACO is executed only on the training set to prevent information leakage. The algorithm searches for a feature subset that maximizes classification performance using an accuracy-based fitness function, following wrapper-based feature-selection principles reported in previous studies [28]. In Scenario B, ACO is applied only to the training set to select an optimal subset of features; the selected subset is then used to train and test the classifiers. The overall ACO workflow and parameter settings are summarized in Table 4:

1. Initialize pheromone levels across all features.
2. Each ant subsequently creates a candidate subset of features.
3. Evaluate subset quality using the classification performance of the target model on validation data, measured by accuracy.
4. Update pheromone trails (evaporation + reinforcement).
5. Stop after N iterations or convergence; output best subset.

Table 4. ACO Parameters

Parameter	Notasi / Field	Nilai
Number of ants	n_ants	10
Number of iterations	n_iterations	30
Pheromone influence	alpha	1.0
Heuristic influence	beta	2.0
Evaporation rate	rho	0.1
Reinforcement constant	q	1.0
Subset size limit	min_features, max_features	5, 15
Cross-validation	cv	3-fold and 5-fold
Objective function	scoring	accuracy
Parameter	Field	value

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Construction of a subset (ant solution). For each ant, a subset size n_select is sampled uniformly from $[min\_features, max\_features]$, and features are selected without replacement according to probabilistic sampling, as shown in Equation 18 to 19 [29] :

$$pi = \frac{(\tau_i)^{\alpha}(\eta_i)^{\beta}}{\sum_{j=1}^{d}(\tau_j)^{\alpha}\,(\eta_j)^{\beta}} \quad (18)$$

where τ_i is pheromone level and $\eta_i$is heuristic importance of feature i. Pheromone update rule.

$$\tau i \leftarrow \tau i + \frac{q.score}{1 + |S|} \quad (19)$$

where $score$ is the CV accuracy of the subset start equation, cap S, and divides cap S is the number of selected features [11].

H. Model Evaluation Metrics

Model performance is assessed using discrimination- and error-based metrics derived from the confusion matrix and the Receiver Operating Characteristic (ROC) curve. Confusion-matrix-based metrics quantify classification correctness at a chosen decision threshold, while ROC–AUC measures the model's ability to separate phishing and legitimate samples across all possible thresholds. This combination provides a comprehensive evaluation of predictive quality, especially for security-related classification tasks where both false positives and false negatives are important. The confusion matrix provides counts for TP, TN, FP, and FN, representing true positives, true negatives, false positives, and false negatives, respectively, while phishing is treated as the positive class [30].

III. RESULT AND DISCUSSION

This section reports the experimental results of five boosting-based classifiers for phishing domain detection and interprets their implications. The main research objective is to (1) identify the best-performing boosting model under a consistent experimental setting and (2) examine whether ACO-based feature selection improves classification performance and/or efficiency. The findings are assessed using quantitative measures such as Accuracy, Precision, Recall, F1-score, and ROC–AUC, complemented by confusion matrices and ROC curves to illustrate the performance of the best models in each scenario [31].

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

A. Results

1. Performance Boosting-Based Models without Feature Selection

In the baseline scenario, Table 4 (all extracted features retained), LightGBM achieves the best overall performance, with 97.24% accuracy and an ROC–AUC of 99.56%, outperforming the other boosting models. This result indicates that LightGBM effectively exploits the complete feature space to model complex interactions among URL/domain attributes. XGBoost, CatBoost, and Gradient Boosting also show strong performance (accuracy > 95%), while AdaBoost records the lowest performance (94.31%), largely due to weaker recall on the phishing class.

Table 5. Result Boosting-Based Models without Feature Selection

Model	ROC-AUC (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
AdaBoost	98.71	94.31	94.47	93.97	94.22
GBM	99.18	95.97	96.00	95.83	95.92
XGBoost	99.57	96.63	96.38	96.81	96.59
CatBoost	99.44	96.63	96.79	96.36	96.58
LightGBM	**99.56**	**97.24**	**97.41**	**96.98**	**97.20**

From the LightGBM confusion matrix, most legitimate domains are correctly classified, whereas misclassifications are more frequent in the phishing class. This pattern is consistent with the tendency of phishing samples to share visual/structural similarities with legitimate domains and reflects the dataset's difficulty in separating borderline cases.

2. Performance Boosting-based Models with ACO Feature Selection

The ACO feature selection process revealed several features consistently retained across both the 3-fold and 5-fold cross-validation configurations, indicating their robustness as phishing indicators. Among the 20 common selected features, key attributes included URL length (length_url), hostname length (length_hostname), number of dots (nb_dots), ratio of digits in the URL (ratio_digits_url), phishing-related keyword count (phish_hints), domain age (domain_age), domain registration length (domain_registration_length), Google index status (google_index), page rank (page_rank), and web traffic (web_traffic). These features are practically relevant because phishing domains commonly employ URL obfuscation techniques, deceptive lexical patterns, recently registered domains, and low-reputation signals, which distinguish them from legitimate websites.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Table 6 summarizes the subset of features selected by the Ant Colony Optimization (ACO) algorithm under the 3-fold cross-validation setting. The selected attributes encompass multiple categories of phishing-related characteristics. These features are identified through the ACO's iterative search mechanism, which generates candidate feature subsets and evaluates them using cross-validation on the training data. During this process, features included in high-performing candidate subsets were more likely to be retained in the final selected subset and were consequently included.

Table 6. Feature Selection Result with 3-fold Cross-Validation

Fold	Attribute Name	Short Description
3	length_url	Total length of the URL
	length_hostname	Length of the domain/hostname
	nb_dots	Number of dots in the URL
	nb_hyphens	Number of hyphens in the URL
	nb_slash	Number of slash characters in the URL
	nb_www	Number of occurrences of “www” in the URL
	ratio_digits_url	Ratio of digits to URL length
	length_words_raw	Total length of words in the URL
	longest_word_path	Length of the longest word in the URL path
	phish_hints	Number of phishing-related keywords
	nb_hyperlinks	Total number of hyperlinks
	ratio_intHyperlinks	Ratio of internal hyperlinks
	ratio_extErrors	Ratio of errors in external links
	links_in_tags	Proportion of links within HTML tags
	safe_anchor	Proportion of safe anchor links
	domain_registration_length	Domain registration duration
	domain_age	Age of the domain
	web_traffic	Indicator of website traffic level
	google_index	Google index status of the domain
	page_rank	Page ranking score of the domain

Table 7 presents the feature subset obtained by applying ACO with a 5-fold cross-validation configuration. The selection of these features results from the ACO optimization procedure, in which feature subsets are repeatedly constructed and assessed via cross-validation, and features associated with higher evaluation scores are reinforced across iterations. Consequently, the final subset reflects attributes that frequently appear in high-scoring candidate solutions during optimization.

Table 7. Feature Selection Result with 5-fold Cross-Validation

Fold	Attribute Name	Short Description
5	length_url	The overall length of the URL
	length_hostname	Overall hostname length
	nb_dots	Number of dots in the URL
	nb_hyphens	Number of hyphens in the URL

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Fold	Attribute Name	Short Description
	nb_slash	Number of slash characters in the URL
	nb_www	Number of occurrences of "www" in the URL
	ratio_digits_url	Ratio of digits to URL length
	length_words_raw	Total length of words in the URL
	longest_word_path	Length of the longest word in the URL path
	phish_hints	Number of phishing-related keywords
	nb_hyperlinks	Total number of hyperlinks
	ratio_intHyperlinks	Ratio of internal hyperlinks
	ratio_extErrors	Ratio of errors in external links
	links_in_tags	Proportion of links within HTML tags
	safe_anchor	Proportion of safe anchor links
	domain_registration_length	Domain registration duration
	domain_age	Age of the domain
	web_traffic	Indicator of website traffic level
	google_index	Google index status of the domain
	page_rank	Page ranking score of the domain
	nb_at	Number of "@" characters in the URL
	nb_qm	Number of question mark ("?") characters in the URL
	shortest_word_host	Length of the shortest word in the hostname
	longest_words_raw	Length of the longest word in the entire URL
	avg_words_raw	Average length of words in the URL

As shown in Table 8, the use of ACO-based feature selection influences the performance of boosting-based classifiers differently across validation settings. For AdaBoost and Gradient Boosting Machine (GBM), the 5-fold configuration yields slight improvements in ROC–AUC, precision, recall, and F1-score compared to the 3-fold setting, while maintaining comparable accuracy. XGBoost consistently achieves the best overall performance under both validation schemes, with accuracy increasing from 97.11% (ROC–AUC = 99.51%) in the 3-fold setting to 97.20% (ROC–AUC = 99.53%) in the 5-fold setting. This result suggests that XGBoost benefits from a more stable feature evaluation and a slightly richer yet informative feature subset, leading to improved generalization.

Table 8. Result Boosting-Based Models with Feature Selection

Model	Fold	Roc-Auc (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
AdaBoost	3	98.17	93.35	93.48	92.82	93.15
	5	98.26	93.35	93.73	92.74	93.23
GBM	3	99.30	96.28	95.25	95.92	95.58
	5	99.41	96.54	95.97	97.08	96.52
XGBoost	3	99.51	97.11	96.90	97.07	96.99

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Model	Fold	Roc-Auc (%)	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
	5	99.53	97.20	97.08	97.25	97.17
CatBoost	3	99.49	96.80	96.54	96.45	96.49
	5	99.53	96.76	96.64	96.81	96.73
LightGBM	3	99.45	96.85	96.20	96.45	96.32
	5	99.52	96.85	96.40	97.25	96.83

CatBoost demonstrates stable performance across both fold settings, with marginal gains in recall and F1-score under the 5-fold configuration, indicating robustness to variations in feature subsets. In contrast, LightGBM exhibits minimal performance variation between the two settings, with accuracy remaining constant at 96.85% while recall and F1-score slightly improve under 5-fold cross-validation. This behavior suggests that LightGBM relies on a broader set of features to construct effective leaf-wise splits, and that aggressive feature reduction does not consistently improve its performance.

The confusion matrix of the best-performing model after feature selection, XGBoost, reveals a balanced error distribution. Specifically, 1,121 legitimate samples are correctly classified with 36 false alarms, while 1,099 phishing samples are correctly identified with 30 missed detections. This balanced misclassification pattern indicates that the model does not exhibit a strong bias toward either class and maintains stable precision–recall characteristics under ACO-based feature selection.

3. Computational Efficiency Comparison Between All Features and ACO-Based Feature Selection

Table 9 presents a computational efficiency comparison between the all-features scenario and the ACO-selected-features scenario in terms of execution time, CPU usage, RAM usage, and classification accuracy. The results indicate that ACO-based feature selection produced mixed effects across the evaluated boosting models. LightGBM benefited from ACO by reducing execution time from 0.467 s to 0.385 s (17.49% faster) and lowering RAM usage, although its accuracy slightly decreased from 97.24% to 96.85%. In contrast, AdaBoost, GradientBoosting, XGBoost, and CatBoost experienced increased execution times after feature selection, despite reductions in RAM usage and slight variations in CPU consumption. Regarding predictive performance, ACO improved accuracy on GradientBoosting, XGBoost, and CatBoost, while slightly reducing performance on AdaBoost and LightGBM. These findings suggest that reducing feature dimensionality does not always directly improve computational efficiency, as model

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

execution is also affected by algorithm-specific optimization mechanisms. Overall, the results demonstrate that the effectiveness of ACO is model-dependent and involves a trade-off between feature compactness, computational efficiency, and predictive performance.

Table 9. Computational Efficiency Comparison Between All Features and ACO-Selected Features

Model	Scenario	Execution Time (s)	CPU Usage (%)	RAM Usage (%)	Accuracy (%)
AdaBoost	All Features	2.249	55.1	12.3	94.31
	ACO Features	4.610	52.9	11.9	93.35
GBM	All Features	5.320	67.4	12.5	95.97
	ACO Features	11.078	65.2	11.9	96.54
XGBoost	All Features	0.656	90.1	12.5	96.63
	ACO Features	0.911	100	11.9	97.20
CatBoost	All Features	0.871	91.3	12.4	96.63
	ACO Features	2.063	98.8	11.9	96.76
LightGBM	All Features	0.467	60.4	12.5	97.24
	ACO Features	0.385	65.5	11.9	96.85

B. Discussion

1. Major Findings and Interpretation

This study confirms that model performance in phishing URL detection is influenced by feature selection: LightGBM performs best in the full-feature setting, while XGBoost benefits more from ACO-based feature reduction.

Two major findings emerge from the experiments:

a. The best-performing model depends on the use of feature selection. Without feature selection, LightGBM achieves the highest performance (97.24% accuracy, 99.56% ROC–AUC). When ACO-based feature selection is applied, XGBoost becomes the top-

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

performing model, achieving 97.20% accuracy and 99.53% ROC–AUC in 5-fold cross-validation, indicating that it benefits more from reduced, more informative feature subsets.

b. ACO is beneficial for some models but not universally. While feature selection improves predictive performance in XGBoost, GradientBoosting, and CatBoost, it slightly reduces the performance of LightGBM and AdaBoost. From a computational perspective, ACO also yields mixed results: LightGBM benefits from faster execution and lower resource usage, whereas other models experience longer execution times despite reduced feature dimensionality. This suggests that feature selection should be treated as model-dependent rather than as a guaranteed means of improving either predictive performance or computational efficiency.

Mechanistically, LightGBM's leaf-wise growth is known to leverage fine-grained feature interactions when many features are available, whereas XGBoost, with regularization and a level-wise growth strategy, often remains robust under feature reduction and can generalize better when redundant attributes are removed [32].

2. Comparison with Previous Research

This study further compares the proposed approach with previous phishing-detection studies that use the same benchmark dataset, as summarized in Table 8. The comparison is conducted based on classification accuracy to ensure a consistent and fair evaluation across different methodologies. Building upon these findings, the proposed method (XGBoost + ACO) in this study further improves classification performance, achieving an accuracy of 97.20%. The results of this research are in line with or supported by previous studies showing that XGBoost is highly effective for phishing URL detection, achieving 96.06% accuracy in earlier work, and that feature selection methods can improve classification performance by reducing redundant features and emphasizing discriminative attributes, which supports the superior performance of the proposed XGBoost + ACO framework in this study [33]. The observed improvement can be attributed to the integration of Ant Colony Optimization for feature selection, which reduces feature redundancy and prioritizes the most discriminative attributes prior to model training. The use of a compact and informative subset of features enables the proposed method to strengthen generalization capability while sustaining strong discriminative performance[29]. Overall, the comparative results indicate that the proposed XGBoost + ACO framework achieves competitive and improved classification performance compared with previous approaches on the same dataset, although the computational effects of feature selection remain model-dependent.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

Table 10. Comparison Previous Research

Reference	Model	Accuracy
Towards Benchmark Datasets For Machine Learning Based Website Phishing Detection: An Experimental Study[34]	Random Forest, Decision tree, Naïve Bayes	91.03 %, 88.11 %, 75.48 %
Improving Web Security through Machine Learning: A Feature-Based Methodology for Detecting Phishing URLs[33]	XGBoost	96.06%
Our propused method	**XgBoost + ACO**	**97.20%**

1. Limitations and Future Work

This study presents several limitations that should be acknowledged when interpreting the reported results. First, the experimental evaluation is conducted using a single phishing dataset, which may limit the generalizability of the findings to other phishing datasets or to real-world traffic distributions with different characteristics. Although the dataset employed in this study contains a comprehensive set of phishing-related features, differences in data sources, feature extraction methodologies, and traffic dynamics may lead to variations in model performance when applied in other contexts.

Second, all models are trained with fixed hyperparameter settings and no systematic hyperparameter optimization procedures, such as grid search, random search, Bayesian optimization, or Optuna, are used. This experimental design is adopted to maintain methodological consistency and reproducibility. However, it implies that the reported performance reflects a controlled experimental setting rather than the optimal configuration of each model. As a result, the performance of the evaluated models may differ across alternative datasets or with more extensive hyperparameter tuning.

Third, the ACO-selected feature subset in this study was optimized using a single benchmark phishing dataset under a fixed experimental setting. Although several selected features represent robust phishing-related indicators, their generalizability to other datasets or to more recent phishing attacks cannot be fully guaranteed. Phishing tactics continually evolve, and attackers may adopt new URL obfuscation patterns, domain manipulation strategies, or webpage structures that reduce the discriminative power of previously effective features. This phenomenon, often referred to as temporal decay of phishing features, may affect the long-term robustness of feature-

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

selection outcomes. Future work should therefore evaluate the stability of ACO-selected features across multiple datasets collected from different periods and investigate adaptive feature-selection approaches that can respond to evolving phishing behaviors.

IV. CONCLUSION

This study examined phishing domain/URL detection using boosting-based machine learning within a controlled, reproducible experimental framework, with the aim of identifying the most effective boosting classifier and assessing the influence of Ant Colony Optimization (ACO) in feature selection across different evaluation settings. A public Kaggle dataset comprising 11,430 samples was used, and a consistent preprocessing pipeline was applied across all experiments. Five boosting classifiers, namely AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost, were evaluated using Accuracy, Precision, Recall, F1-score, and ROC–AUC as performance metrics. The experimental findings show that, in the absence of feature selection, LightGBM achieved the highest overall performance, reaching 97.24% accuracy and a ROC–AUC of 99.56%, indicating its strong ability to effectively utilize the complete engineered feature space. Following the application of ACO-based feature selection, XGBoost emerged as the most consistent top-performing model across both 3-fold and 5-fold cross-validation settings, with the best results obtained under 5-fold cross-validation, yielding 97.20% accuracy and a ROC–AUC of 99.53%. These results indicate that a more stable validation strategy can enhance ACO's effectiveness in selecting informative features. In addition, the findings demonstrate that the impact of feature selection depends on the underlying model: ACO provided measurable benefits for certain boosting classifiers but produced limited or negligible improvements for others, particularly LightGBM. The selected feature subsets highlight the importance of lexical URL characteristics, hyperlink-related indicators, and domain-level attributes as relevant predictors in phishing detection. Overall, this study provides a structured comparative evaluation of boosting-based classifiers and offers empirical evidence regarding the influence of cross-validation strategies on the effectiveness of ACO-based feature selection. Future studies may extend this work by investigating hybrid approaches that incorporate Deep Learning techniques for capturing more complex phishing patterns, assessing model resilience against adversarial phishing scenarios in which malicious URLs are intentionally designed to evade detection, and validating the proposed framework using external as well as temporally shifted datasets to further examine its robustness and generalizability in real-world environments.

Author Contributions. *Kadek Adies Wiranegara*: Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing. *Dandy Pramana Hostiadi*: Supervision, Validation of the Methodology, and Manuscript revision. *Roy Rudolf Huizen*: Investigation, Data Analysis,

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

and Manuscript revision.

All authors have read and agreed to the published version of the manuscript.

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: Data are available upon request from the corresponding author.

Informed Consent: There were no human subjects.

Animal Subjects: There were no animal subjects.

ORCID:
Kadek Adies Wiranegara: https://orcid.org/0009-0000-7199-2332
Dandy Pramana Hostiadi: https://orcid.org/0000-0003-0096-8049
Roy Rudolf Huizen: https://orcid.org/0000-0002-3671-6030

REFERENCES

[1] L. Gallo, D. Gentile, S. Ruggiero, A. Botta, and G. Ventre, "The human factor in phishing: Collecting and analyzing user behavior when reading emails," Comput. Secur., vol. 139, p. 103671, Apr. 2024, doi: 10.1016/j.cose.2023.103671.

[2] S. J. M. S. Al-Naimi, A. Sadighian, and G. Oligeri, "User Behavior Under Phishing Attacks: Eye tracking, Scenarios and Analysis," in 2025 10th International Conference on Information and Network Technologies (ICINT), IEEE, Mar. 2025, pp. 9–16. doi: 10.1109/ICINT65528.2025.11030908.

[3] 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), IEEE, Apr. 2024, pp. 1–17. doi: 10.1109/SEB4SDG60871.2024.10630203.

[4] A. S. I. S. - et al., "Understanding the Impact of Phishing Attacks on Organizational Security and Trust," International Journal For Multidisciplinary Research, vol. 6, no. 6, Dec. 2024, doi: 10.36948/ijfmr.2024.v06i06.34230.

[5] W. Li, S. Manickam, Y.-W. Chong, W. Leng, and P. Nanda, "A State-of-the-Art Review on Phishing Website Detection Techniques," IEEE Access, vol. 12, pp. 187976–188012, 2024, doi: 10.1109/ACCESS.2024.3514972.

[6] D. Komalasari, T. B. Kurniawan, D. A. Dewi, M. Z. Zakaria, Z. Abdullah, and A. Alanda, "Phishing Domain Detection Using Machine Learning Algorithms," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 15, no. 1, pp. 318–327, Feb. 2025, doi: 10.18517/ijaseit.15.1.12553.

[7] H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, and A. Davtyan, "Phishing URL detection with neural networks: an empirical study," Sci. Rep., vol. 14, no. 1, p. 25134, Dec. 2024, doi: 10.1038/s41598-024-74725-6.

[8] S. S. Kumar, P. Muthusamy, and M. P. A. Jerald, "A Hybrid Framework for Improved Weighted Quantum Particle Swarm Optimization and Fast Mask Recurrent CNN to Enhance Phishing-URL Prediction Performance," International Journal of Computational Intelligence Systems, vol. 17, no. 1, Dec. 2024, doi: 10.1007/s44196-024-00663-w.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v10i2.28480

[9] M. A. Daniel, S.-C. Chong, L.-Y. Chong, and K.-K. Wee, "Optimising Phishing Detection: A Comparative Analysis of Machine Learning Methods with Feature Selection," Journal of Informatics and Web Engineering, p., 2025, doi: 10.33093/jiwe.2025.4.1.15.

[10] F. Karimi, M. B. Dowlatshahi, and A. Hashemi, "SemiACO: A semi-supervised feature selection based on ant colony optimization," Expert Syst. Appl., vol. 214, p. 119130, Mar. 2023, doi: 10.1016/j.eswa.2022.119130.

[11] S. Jungjit, A. Prasitsupparote, S. S. Nathan, and T. Rattanakietkhajorn, "Ant Colony Optimization for Multi-Label Correlation-based Feature Selection Method," in 2025 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON), IEEE, Jan. 2025, pp. 797–802. doi: 10.1109/ECTIDAMTNCON64748.2025.10961959.

[12] G. S. Nayak, B. Muniyal, and M. C. Belavagi, "Enhancing Phishing Detection: A Machine Learning Approach With Feature Selection and Deep Learning Models," IEEE Access, vol. 13, pp. 33308–33320, 2025, doi: 10.1109/ACCESS.2025.3543738.

[13] M. K. Prabhakaran, A. D. Chandrasekar, and P. Meenakshi Sundaram, "PHISH_ATTENTION: achieving robust phishing website detection with balanced datasets and advanced URL features," Comput. J., vol. 68, no. 9, pp. 1263–1284, Sep. 2025, doi: 10.1093/comjnl/bxaf036.

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

[15] M. K. Dahouda and I. Joe, "A Deep-Learned Embedding Technique for Categorical Features Encoding," IEEE Access, vol. 9, pp. 114381–114391, 2021, doi: 10.1109/ACCESS.2021.3104357.

[16] M. A. Lones, "Avoiding common machine learning pitfalls," Patterns, vol. 5, no. 10, p. 101046, Oct. 2024, doi: 10.1016/j.patter.2024.101046.

[17] S. Shahidi, A. Wahid Samadzai, and H. Shahbazi, "Effective Data Preprocessing in Data Science: From Method Selection to Domain-Specific Optimization," Journal of Advanced Computer Knowledge and Algorithms, vol. 2, no. 4, pp. 84–90, Jul. 2025, doi: 10.29103/jacka.v2i4.22886.

[18] B. Bala and S. Behal, "A Brief Survey of Data Preprocessing in Machine Learning and Deep Learning Techniques," in 2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), IEEE, Oct. 2024, pp. 1755–1762. doi: 10.1109/I-SMAC61858.2024.10714767.

[19] U. Sarmah, P. Borah, and D. K. Bhattacharyya, "Ensemble Learning Methods: An Empirical Study," SN Comput. Sci., vol. 5, no. 7, p. 924, Oct. 2024, doi: 10.1007/s42979-024-03252-y.

[20] I. G. A. P. Mahendra, I. M. A. Wirawan, and I. G. A. Gunadi, "Enhancement performance of the Naïve Bayes method using AdaBoost for classification of diabetes mellitus dataset type II," International Journal of Advances in Applied Sciences, vol. 13, no. 3, p. 733, Sep. 2024, doi: 10.11591/ijaas.v13.i3.pp733-742.

[21] I. AlShourbaji, N. Helian, Y. Sun, A. G. Hussien, L. Abualigah, and B. Elnaim, "An efficient churn prediction model using gradient boosting machine and metaheuristic optimization," Sci. Rep., vol. 13, no. 1, p. 14441, Sep. 2023, doi: 10.1038/s41598-023-41093-6.

[22] M. ZLOBIN and V. BAZYLEVYCH, "BAYESIAN OPTIMIZATION FOR TUNING HYPERPARAMETRS OF MACHINE LEARNING MODELS: A PERFORMANCE ANALYSIS IN XGBOOST," Computer systems and information technologies, no. 1, pp. 141–146, Mar. 2025, doi: 10.31891/csit-2025-1-16.

[23] P. R, S. Obbayed, K. Manju, G. Swarnalakshmi, and N. Purushotham, "Employee Attrition Prediction based on Light Gradient Boosting Machine with Bayesian Optimization," in 2025 International Conference on Intelligent Systems and Computational Networks (ICISCN), IEEE, Jan. 2025, pp. 1–5. doi: 10.1109/ICISCN64258.2025.10934683.

[24] H. Wu, Y. Mao, J. Weng, Y. Yu, and J. Wang, "Fractional light gradient boosting machine ensemble learning model: A non-causal fractional difference descent approach," Information Fusion, vol. 118, p. 102947, Jun. 2025, doi: 10.1016/j.inffus.2025.102947.

[25] Z. Zeng, "Enhancing Data Science Salary Prediction through CatBoost-Integrated Ensemble Learning: A Comparative Study of Gradient Boosting Methods," in 2025 International Conference on Computers, Information Processing and Advanced Education (CIPAE), IEEE, Aug. 2025, pp. 197–203. doi: 10.1109/CIPAE66821.2025.00040.

[26] S. Rallapalli and Y. M. Mahendra Kumar, "Optimizing Employee Promotion Decisions: A Novel Machine Learning Framework for Predictive Analysis by using GBM CatBoost," in 2024 First International Conference on Software, Systems and Information Technology (SSITCON), IEEE, Oct. 2024, pp. 1–7. doi: 10.1109/SSITCON62437.2024.10796145.

[27] Y. F. Zamzam, T. H. Saragih, R. Herteno, Muliadi, D. T. Nugrahadi, and P. H. Huynh, "Comparison of CatBoost and Random Forest Methods for Lung Cancer Classification using Hyperparameter Tuning Bayesian Optimization-based," Journal of Electronics, Electromedical Engineering, and Medical Informatics, vol. 6, no. 2, pp. 125–136, Apr. 2024, doi: 10.35882/jeeemi.v6i2.382.

[28] M. Ghosh, R. Guha, R. Sarkar, and A. Abraham, "A wrapper-filter feature selection technique based on ant colony optimization," Neural Comput. Appl., vol. 32, no. 12, pp. 7839–7857, Jun. 2020, doi: 10.1007/s00521-019-04171-3.

[29] K. Nemati, A. Sheikhani, S. Kordrostami, and K. Khoshhal, "The embedded feature selection method using ANT colony optimization with structured sparsity norms," Computing, vol. 107, Dec. 2024, doi: 10.1007/s00607-024-01387-7.

[30] T. Reznychenko, E. Uglickich, and I. Nagy, "Categorical Model Estimation with Feature Selection Using an Ant Colony Optimization," in Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics, SCITEPRESS - Science and Technology Publications, 2025, pp. 219–226. doi: 10.5220/0013705300003982.

[31] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," Sci. Rep., vol. 14, no. 1, p. 6086, Mar. 2024, doi: 10.1038/s41598-024-56706-x.

[32] O. Kyrsanov and S. Kryvenko, "Machine learning model for predicting substance properties based on its physicochemical properties," INNOVATIVE TECHNOLOGIES AND SCIENTIFIC SOLUTIONS FOR INDUSTRIES, no. 1(31), pp. 151–165, Mar. 2025, doi: 10.30837/2522-9818.2025.1.151.

[33] R. Alzubi, T. Bishtawi, and H. Kassem, "Improving Web Security through Machine Learning: A Feature-Based Methodology for Detecting Phishing URLs," Engineering, Technology & Applied Science Research, vol. 15, no. 5, pp. 26845–26851, Oct. 2025, doi: 10.48084/etasr.12015.

[34] A. Hannousse and S. Yahiouche, "Towards benchmark datasets for machine learning based website phishing detection: An experimental study," Eng. Appl. Artif. Intell., vol. 104, Sep. 2021, doi: 10.1016/j.engappai.2021.104347.